MERANT™

Edition 6.1

PVCS®
Dimensions™

Database Administrator's Guide

MERANT

3445 NW 211th Terrace

Hillsboro, Oregon 97124

Table of Contents

Welcome to Dimensions

Thank you for choosing MERANT™ PVCS® Dimensions™, a powerful process management and change control system that will revolutionize the way you develop software. Dimensions helps you organize, manage, and protect your software development projects on every level—from storing and tracking changes to individual files, to managing and monitoring an entire development cycle.

Purpose of this manual

This document describes the Dimensions database administration and maintenance utilities used to control the Dimensions databases. The general user of Dimensions does not need this information; it is only required by the Tool Manager who performs these Database Administration functions.

For more information

Refer to the *PVCS Dimensions Getting Started Guide* for a description of the Dimensions Documentation Set, a summary of the ways to work with Dimensions, and instructions for accessing the Online Help.

Edition status

This is Edition *6.1* of the *PVCS® Dimensions™ Database Administrator's Guide.* The information in this edition applies to *Release 7.1 of PVCS® Dimensions™* or later. This edition supersedes earlier editions of this manual.

Acknowledgment

Some of the information contained in this manual regarding the Oracle 8.1.7 Server and Architecture has been reproduced from the Oracle 8 Product Set documentation with permission from Oracle Corporation.

Typographical Conventions

The following typographical conventions are used in the online manuals and online help. These typographical conventions are used to assist you when using the documentation; they are not meant to contradict or change any standard use of typographical conventions in the various Dimensions components or the host operating system.

Convention	Explanation
italics	Introduces new terms that you may not be familiar with and occasionally indicates emphasis.
bold	Emphasizes important information and field names.
UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key.
`monospace`	Indicates syntax examples, values that you specify, or results that you receive.
`monospaced italics`	Indicates names that are placeholders for values you specify; for example, *`filename`*.
`monospace bold`	Indicates the results of an executed command.
vertical rule |	Separates menus and their associated commands. For example, select File | Copy means to select Copy from the File menu. Also, indicates mutually exclusive choices in a command syntax line.
brackets []	Indicates optional items. For example, in the following statement: `SELECT [DISTINCT]`, DISTINCT is an optional keyword.
. . .	Indicates command arguments that can have more than one value.

Ordering Hard-Copy Manuals

As part of your Dimensions license agreement, you may print and distribute as many copies of the PVCS Dimensions manuals as needed.

If you do not want to print each of these online manuals, you can order hard-copy versions from MERANT. To order, please contact your sales representative for assistance.

Contacting Technical Support

MERANT provides technical support for all registered users of this product, including limited installation support for the first 30 days. If you need support after that time, contact MERANT using one of the methods listed in the installation guides, the *Getting Started Guide,* or the Online Help.

Technical support is available 24 hours a day, 7 days a week, with language-specific support available during local business hours. For all other hours, technical support is provided in English.

Support via the web, E-mail, and telephone

SupportNet Customers can report problems and ask questions on the SupportNet web page: `http://support.merant.com/`

To submit an issue, click on the **Report a Problem** link and follow the instructions. You can also submit issues via E-mail or phone. Refer to the installation guides, *Getting Started Guide*, or Online help for a list of contact numbers, including numbers to call for local language support.

The SupportNet Web site contains up-to-date technical support information. Our SupportNet Community shares information via the Web, automatic E-mail notification, newsgroups, and regional user groups.

SupportNet Online is our global service network that provides access to valuable tools and information for an online community for users. SupportNet Online also includes a KnowledgeBase, which contains how-to information and allows you to search on keywords for technical bulletins. You can also download fix releases for your PVCS products.

1 Introduction

In this Chapter

What is Dimensions?

Dimensions is a Process Configuration Management System that supports the development, production and maintenance lifecycles of a product.

A product may easily contain many thousands of hardware, software and documentation items that require management. As the requirements for a product change, and improvements are identified in the production process, the product items will change. Dimensions provides the means to identify, track and control these changes.

How does Dimensions Function?

Dimensions uses a Relational Database Management System (RDBMS) database to control all information relevant to the various aspects of product development. Product items and their various revisions, however, are *not* stored within the database; they are stored as files resident in operating-system directories.

Dimensions makes use of the resources provided by the RDBMS to store information about the products it controls. This information is stored in a RDBMS database, referred to by Dimensions as a base database. Optionally, during an installation process you can create an initial base database PCMS within the RDBMS tablespace *PCMS_DATA*. This base database will contain all the necessary information to get a Dimensions product up and running.

How does Dimensions use the RDBMS?

As mentioned above, Dimensions uses the RDBMS to store information about products currently under its control. This information is held exclusively under RDBMS users called *base databases* to which access rights are selectively granted to various RDBMS users.

Each base database contains a number of schema objects such as tables, constraints, views and indexes. These objects are used to:

- Store information on various Dimensions products.
- Maintain the integrity of this information.
- Allow users selective access to this information.
- Enhance query response times for the various tools accessing this information.

In Dimensions 7.1 or later, when a base database is created, Dimensions also creates an associated user database called `<basedatabase_name>_tool`. This user database is defined to allow registered Dimensions users to access this base database

Dimensions Utilizes Oracle RDBMS

The Dimensions relational database engine is implemented using the SQL-compliant Oracle RDBMS – the relational database developed by Oracle Corporation the well-established market-leader in relational database management technology. The Dimensions delivery medium includes a licensed Oracle subsystem (the Oracle "runtime") whose set up parameters are automatically tailored by the Dimensions installation procedure to values appropriate to the efficient functioning of Dimensions. However, if for any reason the host computer already has an

installation of the *same* version of Oracle as that supplied on the Dimensions delivery medium, then that installation may be used instead (*unless* stated otherwise in the top-level *README* file).

The following subsections give a brief resume of the concepts underlying an Oracle Server. For more detailed information, please consult Oracle's online and hardcopy documentation.

The Oracle Server

The Oracle Server is a relational database management system that provides an open, comprehensive, and integrated approach to information management. An Oracle Server comprises an Oracle database and an Oracle instance. The following sections describe the relationships between the database and the instance.

Structured Query Language (SQL)

SQL is the programming language that defines and manipulates the database. SQL databases are relational databases; this means simply that data is stored as a set of simple relations. A database can have one or more tables, and each table has columns and rows. You can define and manipulate data in a table with SQL commands.

Database Structure

An Oracle database has both a physical and a logical structure. Because the physical and logical server structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

- Physical Database Structure

  An Oracle database's physical structure is determined by the operating system files that constitute the database. Each Oracle database comprises four different types of files:

  - One or more datafiles.
  - Two or more redo log files.
  - One or more control files.
  - *init<db_name/orasid>.ora* initialization files.

  These files provide the actual physical storage for database information.

- Logical Database Structure

  An Oracle database's logical structure is determined by:

  - One or more tablespaces.

    A tablespace is a logical area of storage explained later in this section.

  - The database's schema objects.

    A schema is a collection of objects. Schema objects are the logical structures that directly refer to the database's data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links.

  The logical storage structures, including tablespaces, segments, and extents, dictate how the physical space of a database is used. The schema objects and the relationships among them form the relational design of a database.

An Oracle Instance

Every time a database is started, the *init<orasid>.ora* is read for initialization parameters, a system global area (SGA) is allocated and Oracle background processes are started. The SGA is an area of memory used for database information and is shared by the database users. This combination of background processes and memory buffers is collectively called an Oracle instance.

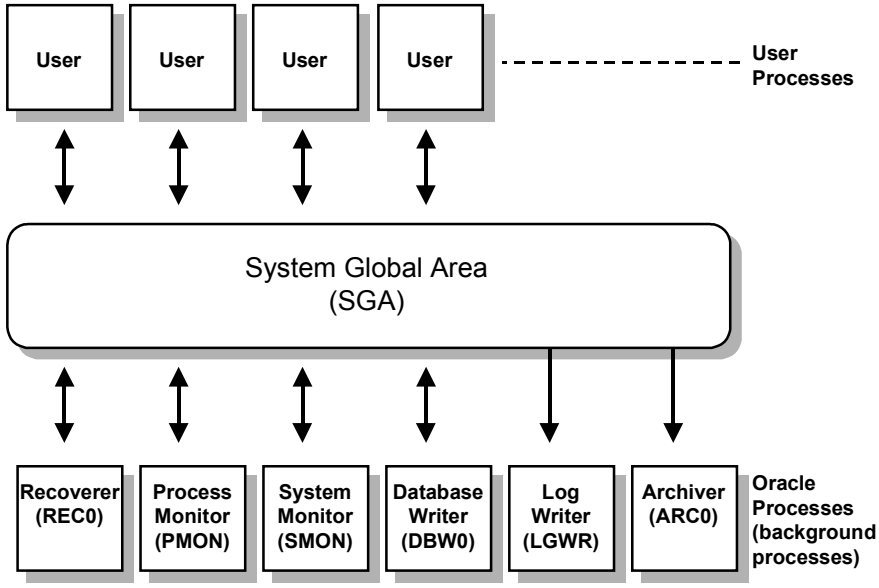

Figure 1-1. Oracle Instance

An Oracle instance has two types of processes: user processes and Oracle processes.

- A user process executes the code of an application program (such as Dimensions) or an Oracle Tool (such as SQL Plus).

- Oracle processes are server processes that either perform work for user processes or are background processes that perform maintenance work for the Oracle Server.

An Oracle System Identifier (SID) is an alphanumeric string used to uniquely identify an instance of Oracle on a computer that may be running multiple Oracle systems concurrently. It must start with an alphabetic character and not exceed the following character sizes: UNIX – 6 and Windows NT/2000 – 4.

In UNIX and Windows NT/2000 installations of Oracle, the database name (<*dbname*>) is always the same as the Oracle instance.

The Oracle NET8 Service Name (called Database Alias on Windows NT/2000) is by default the same as the Oracle SID. It must not exceed eight characters in length. However, if there are other databases in the network with the same Oracle SID, an alternative Service Name should be specified.

Database Structure and Space Management

Relational Database Management Systems

The Relational Database Management System (RDBMS) has three major aspects:

- Structures

  Structures are well defined objects (such as tables, views, indexes, and so on) that store or access the data of a database. Structures and the data contained within them can be manipulated by operations.

- Operations

  Operations are clearly defined actions that allow users to manipulate the data and structures of a database. The operations on a database must adhere to a predefined set of integrity rules.

- Integrity Rules

  Integrity rules are the laws that govern which operations are allowed on the data and structures of a database. Integrity rules protect the data and the structures of a database.

An Oracle database is a collection of data that is treated as a unit. The general purpose of a database is to store and retrieve related information. The database has logical structures and physical structures. An Oracle database can be open (accessible) or closed (not accessible). In normal situations, the database is open and available for use. However, the database is sometimes closed for specific administrative functions that require the database's data to be unavailable to users.

Logical Database Structures

The following subsections explain logical database structures.

- Tablespaces

  A database is divided into logical storage units called tablespaces. A tablespace is used to group related logical structures together. For example, tablespaces commonly group all of an application's objects to simplify certain administrative operations.

  The relationship among databases, tablespaces and datafiles (datafiles are described in the next subsection) is that:

  - Each database is logically divided into one or more tablespaces.

- One or more datafiles are explicitly created for each tablespace to physically store the data of all logical structures in that tablespace.
- The combined size of a tablespace's datafiles is the total storage capacity of that tablespace.
- The combined storage capacity of a database's tablespaces is the total storage capacity of that database.

A tablespace can be online (accessible) or offline (not accessible). A tablespace is normally online so that users can access the information within it. However, sometimes a tablespace may be taken offline to make a portion of the database unavailable while allowing normal access to the remainder of the database. This makes many administrative tasks easier to perform.

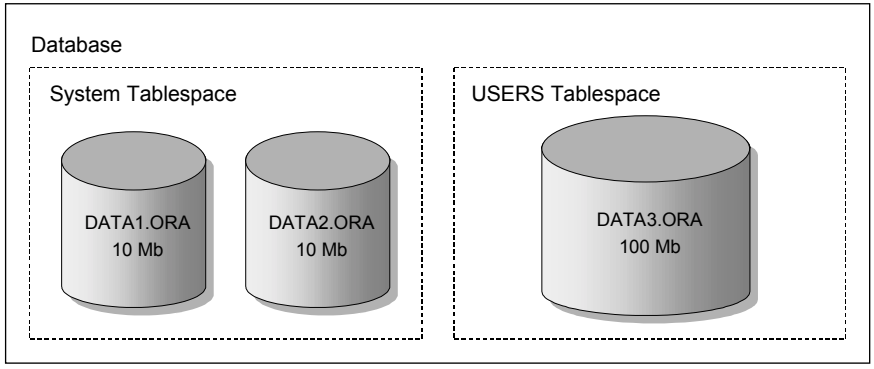

Figure 1-2. Example Database with Tablespaces

■ Schemas and Schema Objects

A schema is a collection of database objects. Schema objects are the logical structures that directly refer to the database's data. Schema objects include such structures as tables, views, sequences, synonyms, indexes, clusters and database links.

- A table is the basic unit of data storage in an Oracle database. The tables of a database hold all of the user-accessible data.
- A view is a custom-tailored presentation of the data from one or more tables. A view can also be thought of as a "stored query". Views do not actually contain or store data; rather, they derive their data from the tables on which they are based.
- A sequence generates a serial list of unique numbers for numeric columns of a database's tables. Sequence numbers are independent of tables, so the same sequence can be used for one or more tables.
- A synonym is an alias for a table, view or sequence. A synonym is not actually an object itself, but is instead a direct reference to an object. Synonyms are used to:
    - mask the real name and owner of an object,
    - provide public access to an object,
    - provide location transparency for the tables and views held in remote databases,
    - simplify the SQL statements for database users.
- Indexes, clusters, and hash clusters are optional structures associated with tables, which can be created to increase the performance of data retrieval.
    - Indexes are created to increase the performance of data retrieval. Indexes are logically and physically independent of the data.
    - Clusters are an optional method of storing table data. Clusters are groups of one or more tables physically stored together because they share common columns.
    - Hash clusters also cluster table data in a manner similar to normal, index clusters. However, a row is stored in a hash cluster based on the result of applying a hash

function to the row's cluster key value. All rows with the same hash key value are stored together on disk.

- A database link is a named object that describes a "path" from one database to another. Database links are implicitly used when a reference is made to a global object name in a distributed database.

■ Data Blocks, Extents, and Segments

Oracle allows fine-grained control of disk space usage through the logical storage structures, including data blocks, extents, and segments.

■ *Oracle Data Blocks.* At the finest level of granularity, an Oracle database's data is stored in data blocks. One data block corresponds to a specific number of bytes of physical database space on disk. A data block size is specified for each Oracle database when the database is created. A database uses and allocates free database space in Oracle data blocks.

■ *Extents.* The next level of logical database space is called an extent. An extent is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information.

■ *Segments.* The level of logical database storage above an extent is called a segment. A segment is a set of extents allocated for a certain logical structure. For example, the different types of segments include the following:

- *Data Segment*. Each non-clustered table has a data segment. All of that table's data is stored in the extents of its data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.

- I*ndex Segment.* Each index has an index segment that stores all of its data.

- *Rollback Segment.* One or more rollback segments are created by the database administrator for a database to temporarily store "undo" information. This information is used:
  - to generate read-consistent database information,
  - during database recovery,
  - to rollback uncommitted transactions for users.
- *Temporary Segment*. Temporary segments are created by Oracle when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the temporary segment's extents are returned to the system for future use.

Oracle dynamically allocates space when the existing extents of a segment become full. Therefore, when the existing extents of a segment are full, Oracle allocates another extent for that segment as needed. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

Physical Database Structures

The following subsections explain the physical database structures of an Oracle database.

■ Datafiles

Every Oracle database has one or more physical datafiles. A database's datafiles contain all the database data. The data of logical database structures such as tables and indexes is physically stored in the datafiles allocated for a database.

■ The Use of Datafiles

The data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle. Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and

increase performance, data is pooled in memory and written to the appropriate datafiles all at once, as determined by Oracle background processes.

- Redo Log Files

  Every Oracle database has a set of two or more redo log files. The set of redo log files for a database is collectively known as the database's redo log. The primary function of the redo log is to record all changes made to data. Should a failure prevent modified data from being permanently written to the datafiles, the changes can be obtained from the redo log and work is never lost. Redo log files are critical in protecting a database against failures. To protect against a failure involving the redo log itself, Oracle allows a multiplexed redo log so that two or more copies of the redo log can be maintained on different disks.

- The Use of Redo Log Files

  The information in a redo log file is used only to recover the database from a system or media failure that prevents database data from being written to a database's datafiles. For example, if an unexpected power outage abruptly terminates database operation, data in memory cannot be written to the datafiles and the data is lost. However, any lost data can be recovered when the database is opened, after power is restored. By applying the information in the most recent redo log files to the database's datafiles, Oracle restores the database to the time at which the power failure occurred.

- ARCHIVELOG Mode

  When a database's redo log is used in ARCHIVELOG mode, the Oracle instance’s ARCH (Archive task) background process becomes active. This process copies the online redo log files to archival storage when they become full.

- Control Files

  Every Oracle database has a control file. A control file contains entries that specify the physical structure of the database. For example, it contains the following types of information:

  - database name
  - names and locations of a database's datafiles and redo log files
  - time stamp of database creation.

  Like the redo log, Oracle allows the control file to be multiplexed for protection of the control file.

- The Use of Control Files

  Every time an instance of an Oracle database is started, its control file is used to identify the database and redo log files that must be opened for database operation to proceed. If the physical makeup of the database is altered (for example, a new datafile or redo log file is created), the database's control file is automatically modified by Oracle to reflect this change.

  A database's control file is also used if database recovery is necessary. Although only one control file is required, it is advisable to have more than one control file per Oracle instance, preferably on different disks. In such circumstances, Oracle automatically ensures that each control file is identical to each other control file.

The Data Dictionary

Each Oracle database has a data dictionary. An Oracle data dictionary is a set of tables and views that are used as a read-only reference about the database. For example, a data dictionary stores information about both the logical and physical structure

of the database. In addition to this valuable information, a data dictionary also stores such information as:

- The valid users of an Oracle database.
- Information about integrity constraints defined for tables in the database.
- How much space is allocated for a schema object and how much of it is being used.

A data dictionary is created when a database is created. To accurately reflect the status of the database at all times, the data dictionary is automatically updated by Oracle in response to specific actions (such as when the structure of the database is altered). The data dictionary is critical to the operation of the database, which relies on the data dictionary to record, verify, and conduct ongoing work. For example, during database operation, Oracle reads the data dictionary to verify that schema objects exist and that users have proper access to them.

Oracle Server Architecture

The following section discusses the memory and process structures used by an Oracle Server to manage a database.

Memory Structures

Several basic memory structures are associated with Oracle:

- System Global Area (SGA)

  The SGA is a shared memory region that contains data and control information for one Oracle instance. An SGA and the Oracle background processes constitute an Oracle instance.

  Oracle allocates the SGA when an instance starts and deallocates it when the instance shuts down. Each instance has its own SGA.

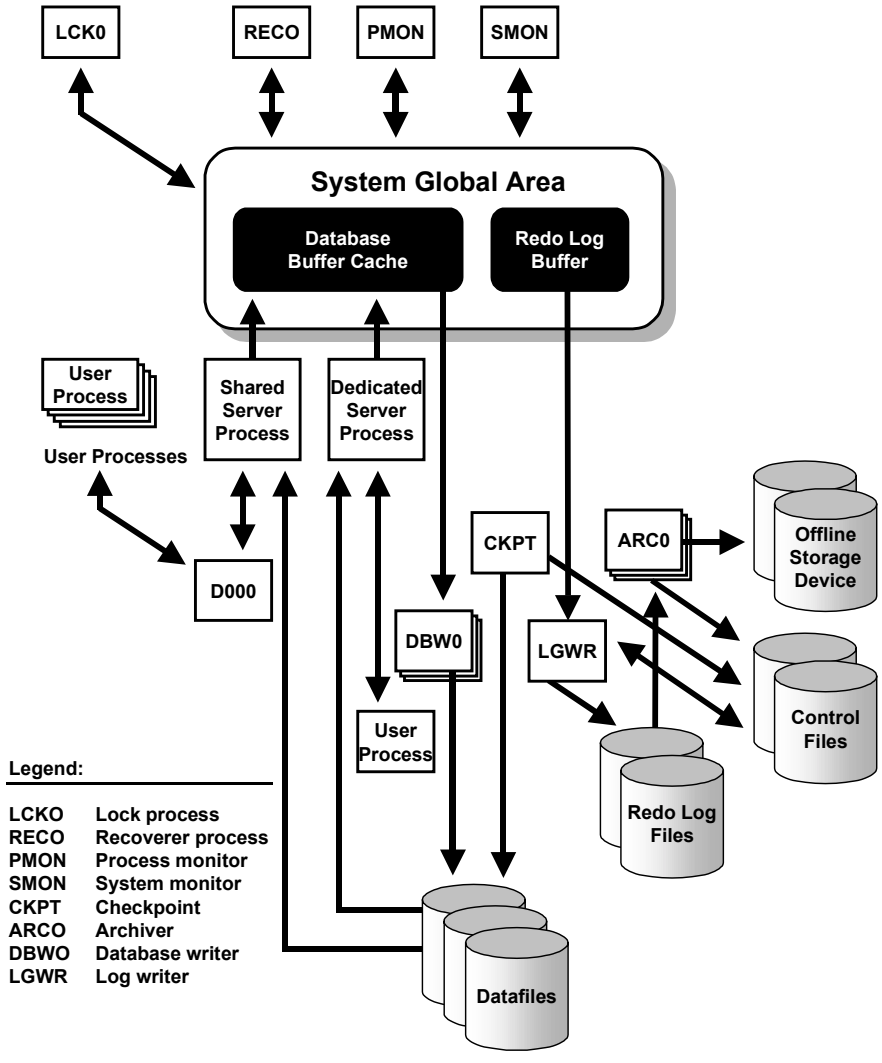

Figure 1-3. Oracle Server Memory and Process Structure Interactions

Users currently connected to an Oracle Server share the data in the SGA. For optimal performance, the entire SGA should

be as large as possible (while still fitting in real memory) to store as much data in memory as possible and minimize disk I/O. The information stored within the SGA is divided into several types of memory structures, including the database buffers, redo log buffer, and the shared pool. These areas have fixed sizes and are created during instance startup.

- *Database Buffer Cache*. Database buffers of the SGA store the most recently used blocks of database data; the set of database buffers in an instance is the database buffer cache. These buffers can contain modified data that has not yet been permanently written to disk. Because the most recently (and often the most frequently) used data is kept in memory, less disk I/O is necessary and performance is increased.
- *Redo Log Buffer*. The redo log buffer of the SGA stores redo entries–a log of changes made to the database. The redo entries stored in the redo log buffers are written to an online redo log file, which is used if database recovery is necessary. Its size is static.
- *Shared Pool.* The shared pool is a portion of the SGA that contains shared memory constructs such as shared SQL areas. A shared SQL area is required to process every unique SQL statement submitted to a database. A single shared SQL area is used by multiple applications that issue the same statement, leaving more shared memory for other uses.
- *Cursors.* A cursor is a handle (a name or pointer) for the memory associated with a specific statement. Most Oracle users rely on the automatic cursor handling of the Oracle utilities.
- *Program Global Area (PGA).* The PGA is a memory buffer that contains data and control information for a server process. A PGA is created by Oracle when a server process is started. The information in a PGA depends on the configuration of Oracle.

Processes

An Oracle Server has two general types of processes: user processes and Oracle processes.

- User (Client) Processes

  A user process is created and maintained to execute the software code of an application program or an Oracle tool. The user process also manages the communication with the server processes. User processes communicate with the server processes through the program interface.

- Oracle Processes

  Oracle processes are called by other processes to perform functions on behalf of the invoking process. The different types of Oracle processes comprise:

  - *Server Processes*. Oracle creates server processes to handle requests from connected user processes. A server process is in charge of communicating with the user process and interacting with Oracle to carry out requests of the associated user process. Oracle can be configured to vary the number of user processes per server process. In a dedicated server configuration, a server process handles requests for a single user process. A multi-threaded server configuration allows many user processes to share a small number of server processes, minimizing the number of server processes and maximizing the utilization of available system resources.

  - *Background Processes*. Oracle creates a set of background processes for each instance. These consolidate functions that would otherwise be handled by multiple Oracle programs running for each user process. The background processes asynchronously perform I/O and monitor other Oracle processes to provide increased parallelism for better performance and reliability.

An SGA and the Oracle background processes constitute an Oracle instance. Each Oracle instance may use several background processes. The names of these processes are *DBWn*, *LGWR*, *CKPT*, *SMON*, *PMON*, *ARCn*, *RECO*, *Dnnn*, *LCK0*, *SNPn* and *QMNn* (the last five are not supported in the Dimensions Oracle runtime).

- *Database Writer (DBWn)*. The *DBWn* writes modified blocks from the database buffer cache to the datafiles. Although one database writer process (*DBW0*) is sufficient for most systems, you can configure additional processes (*DBW1* through *DBW9*) to improve write performance for a system that heavily modifies data. The initialization parameter *DB_WRITER_PROCESSES* specifies the number of *DBWn* processes.

  Since Oracle uses write-ahead logging, *DBWn* does not need to write blocks when a transaction "commits". Instead, *DBWn* is designed to perform batched writes with high efficiency. In the most common case, *DBWn* writes only when more data needs to be read into the system global area and too few database buffers are free. The least recently used data is written to the datafile first. *DBWn* also performs writes for other functions such as checkpointing.

- *Log Writer (LGWR)*. The *LGWR* writes redo log entries to disk. Redo log data is generated in the redo log buffer of the SGA. As transactions commit and the log buffer fills, *LGWR* writes redo log entries into online redo log files.

- *Checkpoint (CKPT)*. At specific times, all modified database buffers in the SGA are written to the datafiles by *DBWn*; this event is called a checkpoint. The *CKPT* process is responsible for signaling *DBWn* at checkpoints and updating all the datafiles and control files of the database to indicate the most recent checkpoint.

- *System Monitor (SMON)*. *SMON* performs instance recovery at instance startup. *SMON* also cleans up temporary segments that are no longer in use and recovers dead transactions skipped during crash and instance recovery because of file-read or offline errors. These transactions are eventually recovered by *SMON* when the tablespace or file is brought back online. *SMON* also coalesces free extents within the database to make free space contiguous and easier to allocate.
- *Process Monitor (PMON)*. *PMON* performs process recovery when a user process fails. *PMON* is responsible for cleaning up the cache and freeing resources that the process was using. *PMON* also checks on dispatcher (see below) and server processes and restarts them if they have failed.
- *Archiver (ARCn)*. The archiver copies the online redo log files to archival storage when they are full or a log switch occurs. Although a single *ARCn* process (*ARC0*) is sufficient for most systems, you can specify up to ten *ARCn* processes by using the dynamic initialization parameter *LOG_ARCHIVE_MAX_PROCESSES*. If the workload becomes too great for the current number of *ARCn* processes, *LGWR* automatically starts another process (up to the maximum of ten). *ARCn* is active only when a database is in *ARCHIVELOG* mode and automatic archiving is enabled.
- *Recoverer (RECO) [Not supported in Dimensions Oracle runtime]*. *RECO* is used to resolve distributed transactions that are pending due to a network or system failure in a distributed database. At timed intervals, the local *RECO* attempts to connect to remote databases and automatically complete the commit or rollback of the local portion of any pending distributed transactions.
- *Dispatcher (Dnnn) [Not supported in Dimensions Oracle runtime]*. Dispatchers are optional background processes, present only when a multi-threaded server configuration is used. At least one dispatcher process is

created for every communication protocol in use (*D000, … Dnnn*). Each dispatcher process is responsible for routing requests from connected user processes to available shared server processes and returning the responses back to the appropriate user processes.

- *Lock (LCK0) [Not supported in Dimensions Oracle runtime]*. The lock process (*LCK0*) is used for inter-instance locking when the Oracle Parallel Server is used.
- *Job Queue (SNPn) [Not supported in Dimensions Oracle runtime].* In a distributed database configuration, up to 36 job queue processes (*SNP0, ..., SNP9, SNPA, ..., SNPZ*) can automatically refresh table snapshots. These processes wake up periodically and refresh any snapshots that are scheduled to be automatically refreshed. If more than one job queue process is used, the processes share the task of refreshing snapshots. These processes also execute job requests created by the *DBMS_JOB* package and propagate queued messages to queues on other databases.
- *Queue Monitor (QMNn) [Not supported in Dimensions Oracle runtime]*. Queue monitors are optional background processes that monitor the message queues for Oracle Advanced Queuing (Oracle AQ). You can configure up to ten queue monitor processes.

What is a DBA and their Responsibilities?

The Database Administrator (DBA) is responsible for maintaining the overall RDBMS database and has responsibility to:

- Start up and shut down the RDBMS[1]. Note that the Oracle Service Manager Utility with the *svrmgrl* command (UNIX and Windows NT/2000) provides alternative functions that can be used for administrative purposes.
- Take regular image backups of the RDBMS database files and item libraries as well as exports of the entire RDBMS database and/or base databases (see "EXP – Oracle Export Utility" on page 78, "Full System Exports" on page 128 and "IMP – Oracle Import Utility" on page 80). These steps are essential to ensure minimal damage in the event of database corruption caused by media failure.
- Decide on the possible use of Archivelog mode (contact PVCS Support Center for more information).
- When necessary, increase the size of the RDBMS database to provide more space for Dimensions (see "INCR – Increase the RDBMS Database Space" on page 62).
- Ensure that the disks hosting the item libraries have an adequate allocation of free space.
- Ensure that the RDBMS has sufficient space before a new base database is created (see "SPAC – Display RDBMS Database Space Usage" on page 57).
- Monitor the performance of the RDBMS with the view of identifying areas of performance improvement for Dimensions (see Chapter 7, "Dimensions Quick Tuning Guide").
- Reorganize the RDBMS database periodically to reclaim the space unavailable due to fragmentation (see "SPAC – Display RDBMS Database Space Usage" on page 57).

---

1. For the UNIX version of Dimensions, the installation process creates the necessary startup and shutdown scripts in the *PCMS_MMI* directory.

- Change database passwords when required (see "CPAS – Change Dimensions Database Password" on page 98). However, there is a facility for "secure" databases in which the passwords are unknown to the user and therefore never require changing.
- Create base databases (see "CRDB – Create Dimensions Base Database" on page 70) and assign Tool Managers to manage them (see "CTM – Change Tool Manager" on page 101).
- Delete any unwanted base databases or user databases (see, respectively, "DLDB – Delete Dimensions Base Database" on page 74 or "DUSR – Delete Dimensions User Database" on page 96).
- Create user databases (see "UREG – Register Users" on page 105).

What are the Responsibilities of a Tool Manager?

When the DBA creates a Dimensions base database, an operating-system account is assigned to act as the Tool Manager for that base database.

The Tool Manager has the responsibility to:

- Create and delete user databases.
- Change database passwords when required.

In order to perform the above operations, the Tool Manager must be registered by the DBA as a Dimensions user on that base database.

Notes

1 A DBA will also have the role of a Tool Manager on a base database, in which case the above activities will overlap. As this is generally expected to be the case, the terms "DBA" and "Tool Manager" are used interchangeably in the rest of this document unless stated otherwise.

2 On UNIX versions of Dimensions, for future convenience all operating-system accounts set up as Tool Managers should also be placed in the same DBA group. Such Tool Managers will then be able to perform Dimensions Network operations using the “net_admin” utility.

What DBA Operations are Provided?

The full suite of Dimensions-supported DBA operations is invoked through the use of the command line interface (CLI) of Dimensions. Some of these utilities are also available via the graphical user interface (GUI) of the Dimensions Process Modeler. Maintenance and general database operations that may be required in addition to those supported by Dimensions (e.g. to perform a full system export) are invoked through the use of standard Oracle utilities. These utilities are described in “Access to the DBA Utilities” on page 46.

The DBA and Tool-Manager facilities of Dimensions enable the Database Administrator or Tool Manager to:

- Monitor database space usage and increase it when necessary.
- Define and delete base databases. (Creation, only, is also available via the Process Modeler.)
- Define and drop Dimensions users. (Also available via the Process Modeler.)

- Update details on Dimensions users and passwords. (Updating user details, except for passwords, is also available via the Process Modeler.)

The majority of DBA activities are divided into the following main areas:

- RDBMS Maintenance Procedures.
- Dimensions Database Administration Procedures.
- Dimensions User Administration Procedures.

The following chapters describe each of these areas in detail.

Notes

1 Both Dimensions and Oracle must have been installed prior to running any DBA utilities. The installation procedures are detailed in the appropriate operating-system version of the relevant PVCS® Dimensions™ Installation Guide.

2 This document applies to UNIX and Windows NT/2000 operating systems; operating-system specifics are marked as such. Take care to use the correct syntax for the specific operating system in use, in particular UNIX is case-sensitive and has different file naming conventions from Windows NT/2000. In some of the UNIX specific examples a variable called <*db_name*> is used. This will be the same as <*orasid*> if the database was created as part of the Dimensions installation.

Types of Database User

Dimensions uses several different kinds of database account. These include the following:

- Base database account

- Registered user account
- Unregistered user database account
- Reporter user database account
- *PCMS_SYS* account.

These account types are discussed in the following sections.

NOTE For brevity, within the following sections the terms *CRDB* and *UREG* should be understood to include their Process Modeler equivalents unless explicitly stated otherwise.

To ensure security of the database, most of the Dimensions database accounts may be created to use either plain text or encrypted passwords. These security considerations are discussed below.

Security Considerations

For the security of the RDBMS, any changes to its constituent information must be suitably controlled. Additionally, for the proper functioning of Dimensions, it is important that changes to the Dimensions database *must* only be made through Dimensions itself.

The *base database* is allocated a *plain* text password by default. This is in order to provide for one route into the Dimensions database for any maintenance activities, and to allow additional DBA functions to have access to the Dimensions database. DBA and Tool Managers are advised to change this password as soon as practicable, and to treat the password as they would a root or system manager password.

The *reporter user* needs to have a *plain* text password to enable third-party tools to access the base database tables. Database protection for this type of user is achieved by Dimensions

ensuring that the reporter user has only *read* access to the tables via a set of Dimensions Published Views.

The *unregistered (proxy) user* facility is provided in order to allow large numbers of *occasional* users to enter change documents without the overhead of setting them up through the *CUSR* function. These users are prevented from accessing functions outside of the change management area.

Base Database Account

CRDB (see "CRDB – Create Dimensions Base Database" on page 70) creates a base database containing a full set of tables and views used to store product (project) data. If a plain text password is assigned to the base database when it is created (via the */NOSECURE* parameter), the Tool Manager is advised to change this password as soon as practicably possible and to treat it as they would a root password. The *CRDB* command also creates a user database for the Tool Manager to allow them to maintain the base database.

Registered User Account

UREG (see "UREG – Register Users" on page 105) registers a user that enables select, update, insert and delete access to the tables and views in the base database.

Unregistered User Database Account

CUSR with the qualifier */PROXY* (or Process Modeler Account Type "Proxy") creates a user database that enables, insert, update and delete access to a limited range of Dimensions functions.

Any user may enter Dimensions on this type of account without previous registration. The user's login identity is recorded by Dimensions on the first access. This allows Dimensions to track change documents to and from that user, and also allows the Tool Manager subsequently to enter user details such as location and telephone number (by the *PROF* function, see "PROF – Display User's Profile Information" on page 101).

The limited range of Dimensions functions comprises:

- Creating change documents (for any product).
- Actioning change documents in *$ORIGINATOR* role.
- Browsing change documents if authorized by the applicable product manager.
- Generating standard change management reports if authorized by the applicable product manager.
- Performing query and report functions provided in other areas of Dimensions.

The authorization referenced above is granted to all users of this type when the Product Manager for the product concerned assigns the role *$PCMS-CM-USER* to the wild card user "*".

Reporter User Database Account

CUSR with the qualifier */REPORT* (or Process Modeler Account Type "Report") creates a user database that enables report writers and other third-party products to access the Dimensions database in read-only mode. Access is given to Dimensions Published Views but no access to Dimensions functionality is allowed, nor is this type of user given direct access to the tables and views of the base database. Refer to the *Reports Guide* for further information on how to set up these users.

PCMS_SYS Account

This account owns tables containing data applicable to the entire Dimensions installation. This includes details on the network configuration used by Dimensions Network and the installation identification information.

This account is created during installation (or via the *fill_pcms_sys* script) and has a plain text password. The *PCMS_SYS* account *cannot* be used to access Dimensions: it acts as an adjunct to the entire base databases in a single Dimensions installation.

2 General Operating Guide

In this Chapter

This chapter of the guide describes how to access the Dimensions and Oracle DBA utilities via the various Command Line Interfaces (CLIs) and Graphical User Interfaces (GUIs).

General Prerequisites

It is essential that the user performing the DBA role becomes familiar with the conventions of the Dimensions DBA utilities in order to fully exploit their capabilities.

Access to the DBA functions from within Dimensions requires the user to be a registered Dimensions DBA user. Initially the installation will set this up for *pcms*, or the login-id of the account into which Dimensions has been installed. The Process Modeler function *User Registration | (Tool Manager) | Add* can be used to empower users other than *pcms* to use these DBA functions. These utilities allow a user such as the Tool Manager for a new base database to be the DBA for that database, enabling that user to use the Dimensions DBA utilities.

To enable a DBA user to run DBA utilities, Dimensions first needs to set them up as a Dimensions user with an appropriate operating-system environment. Dimensions creates this environment by automatically invoking the following operating-specific installation/configuration files:

- *pcmslogin* or *pcmsprofile* on UNIX
- *pcms.cfg* on Windows NT/2000

These files are described in the PVCS Dimensions Installation Guide appropriate to each operating system.

In addition, the DBA user will also need to meet further operating-system-specific prerequisites as described in the following subsections.

UNIX Prerequisites

The UNIX account from which certain DBA operations are to be performed must be a member of the group *dba* as defined in the file */etc/group.* This will automatically enable the *Service Manager* and *sqlplus* utilities.

Also, with both the Tool Manager's UNIX account and Dimensions' UNIX account (*pcms*) being in the same group (*dba*), the Tool Manager will be able to access the Dimensions Network "net_admin" utilities.

Windows NT/2000 Dimensions Server Prerequisites

The Windows NT/2000 account from which the DBA is to be run, normally *pcms*, has to be an Oracle *OPS$* user in order to perform certain functions e.g. increasing database space. Following an installation of Dimensions on Windows NT/2000, this *OPS$* user will already been created for the owner of the Dimensions installation. Being an Oracle *OPS$* user will give the DBA the privilege to perform Oracle maintenance operations through the *svrmgr* and *sqlplus* utilities.

NOTE A Dimensions server can be installed on any Windows NT/2000 platform that meets the requirements stated in the *README* file This includes *both Server and Workstation* versions of Windows NT/2000.

Windows 98 Dimensions Tools Prerequisites

If the “Dimensions Tools” have been installed, the DBA will be able to perform Dimensions DBA utilities, via a remote login to the Dimensions server, utilizing:

- PC Client’s “Run” CLI.
- The Process Modeler.

If the “Oracle Client Tools for Dimensions” have been installed, the DBA will be able to perform Oracle maintenance operations through the *sqlplus* utility, specifying the remote Oracle Service Name.

Access to the DBA Utilities

The facilities provided by Dimensions and Oracle can only be used after the above prerequisites have been met. The DBA functions are limited to DBAs only.

NOTE A DBA will have the role of a Tool Manager for a base database. As this is generally expected to be the case, the terms "DBA" and "Tool Manager" are used interchangeably unless stated otherwise.

Access to Dimensions DBA Utilities

The Dimensions DBA utilities can be accessed via various CLIs and GUIs as shown in the following table:l

Operating System	Command Line Interfaces	Graphical User Interfaces
UNIX	■ `pcms <command>` at the operating-system prompt.[a] ■ `pcms xcmd <filename>` at the operating-system prompt.[a]	Process Modeler [b] launched from Motif Client's Utilities menu or `pcms_pm` at the operating-system prompt.
Windows NT/2000	■ `pcms <command>` at the NT/2000 command prompt. [a] ■ `pcms xcmd <filename>` at the NT/2000 command prompt.[a] ■ `cmdclient <command>` at the prompt in the *Command Prompt* CLI of NT/2000. [c] ■ `cmdclient xcmd <filename>` at the NT/2000 command prompt. [c] ■ Submitting `<command>` or `xcmd <filename>` at the NT/2000 command prompt. [c]	*Process Modeler* [b] launched from/by: ■ the *PVCS Dimensions* menu from the *Programs* menu from the NT or 2000 *Start* button. ■ *ifrun60 process_wizard* at the at the NT/2000 command prompt.

Note: *Explanations of Footnotes [a] to [c] are given on the next page*

Operating System	Command Line Interfaces	Graphical User Interfaces
Windows 98	■ `cmdclient <command>` at the MS-DOS prompt. [c] [d] ■ `cmdclient xcmd <filename>` at the MS-DOS prompt. [c] [d] ■ Submitting `<command>` `or xcmd <filename>` in the *Run* CLI of PC Client. [c] [d]	*Process Modeler* [e] launched from/by: ■ the *PVCS Dimensions* menu from the *Programs* menu from the 98 *Start* button, ■ *ifrun60 process_wizard* at the MS-DOS Prompt.

a. Dimensions commands provide the majority of DBA utilities.
b. The Process Modeler provides a subset of typical common DBA utilities, by accessing the Process Modeler GUI-based forms functions *"Export Process Model"*, *"Create New Base Database"* and *"User Registration"*. Refer to *Process Modeling User's Guide* for more detail.
c. Dimensions commands provide the majority of DBA utilities.
d. As a server cannot be installed on a Windows 98 platform, PC Client (and by association *cmdclient*) on such a platform must always access a *remote* server. Consequently, any listings or logs generated by DBA commands thus invoked are deposited in the home directory of the username being used on the node hosting the server.
e. The Process Modeler provides a subset of typical common DBA utilities, by accessing the Process Modeler GUI-based forms functions *"Export Process Model"*, *"Create New Base Database"* and *"User Registration"*. Refer to *Process Modeling User's Guide* for more detail.

Access to Oracle DBA Utilities

Maintenance and general database operations that may be required in addition to those supported by Dimensions (e.g. to perform a full system export) can be accessed via the operating system prompt and Oracle's standard CLIs as shown in the table

below. (Note that there are no GUIs for accessing these operations.)

Operating System	Command Line Interfaces
UNIX	■ Commands such as *exp* and *imp* at the operating-system prompt. ■ *svrmgrl* at the operating-system prompt. ■ *sqlplus* at the operating-system prompt.
Windows NT/ 2000	■ Commands such as *exp* and *imp* at the operating-system prompt. ■ *svrmgrl* at the NT/2000 command prompt. ■ *sqlplus* at the NT/2000 command prompt. ■ Submitting *sqlplus* commands in the *SQL Plus* CLI launched from the *PVCS Dimensions Oracle* menu from the *Programs* menu from the NT or 2000 *Start* button.
Windows 98	■ *sqlplus* at the prompt in the *MS-DOS Prompt* CLI of 98.[a] ■ Submitting *sqlplus* commands in the *SQL Plus* CLI launched from the *Dimensions Oracle* menu from the *Programs* menu from 98's *Start* button. [a]

a. As an Oracle database cannot be installed on a Windows 98 platform, SQL*Plus on such a platform must always access a *remote* database via the Oracle service name mechanism.

Dimensions DBA Command Execution

UNIX Execution

The manner in which Dimensions DBA “commands” are submitted to the operating system for execution depends on the interface from which they are entered:

- If the “command” is entered as a Dimensions DBA command at the operating-system prompt, then the command will be run as a background task. The user will be queried as to whether the task (job) should be run now or sometime in the future:

  - If the user answers *yes*, then the command will be run immediately as a background task and when it completes a mail message containing a log of the command's execution will be sent to the user. This mail message should be examined to determine whether the job was successful.

  - If the user answers *no*, the command script specified as the job name will be left in the user's current directory and can be executed at a later date using standard operating-system batch utilities.

  An exception to the above is *CPAS* (Change Dimensions Database Password), which runs in the foreground.

- If the “command” is entered via the Process Modeler's GUI-based forms, then the “command” will be run immediately as a synchronous task. The Process Modeler's functions applicable to DBA operations are *“Export Process Model”*, *“Create New Base Database”* and *“User Registration”.*

Windows NT/2000 Execution

The manner in which Dimensions DBA "commands" are submitted to the operating system for execution depends on the interface from which they are entered:

- If the "command" is entered as a Dimensions DBA command in the Command Prompt CLI, then the command will be run as a background task. The user will be queried as to whether the task (job) should be run now or sometime in the future:

  - If the user answers *yes*, then the command will be run immediately as a background task and when it completes a mail message containing a log of the command's execution will be sent to the user. This mail message should be examined to determine whether the job was successful.

  - If the user answers *no*, the command script specified as the job name will be left in the user's current directory and can be executed at a later date using standard operating-system batch utilities.

  An exception to the above is *CPAS* (Change Dimensions Database Password), which runs in the foreground.

- If the "command" is entered as a *CMDCLIENT* DBA command in the Command Prompt CLI, then the command will be run immediately as a background task (the user will *not* be queried as to whether the task should be run now or sometime in the future). When it completes a mail message containing a log of the command's execution will be sent to the user; this mail message should be examined to determine whether the job was successful.

  It should be remembered that *CMDCLIENT* "talks" to the Dimensions server specified in the login of the active associated PC Client. Therefore, if a "remote" (rather than "local") login was specified, then the following is delivered

to the Dimensions server using the *username specified in the PC Client remote login procedure:*

- The mail message described above.
- Any listings or logs generated by the DBA commands (these will be deposited in the home directory of the username being used on the Dimensions server).

■ If the "command" is entered as a Dimensions DBA command via PC Client's *Run* CLI, then the command will be run immediately as a background task (the user will *not* be queried as to whether the task should be run now or sometime in the future). The execution behavior for a PC Client remote login regarding mail and output listings or logs is the same as that described for `CMDCLIENT` above.

■ If the "command" is entered via the Process Modeler's GUI-based forms, then the "command" will be run immediately as a synchronous task. The Process Modeler's functions applicable to DBA operations are *"Export Process Model"*, *"Create New Base Database"* and *"User Registration".*

Windows 98 Execution

The manner in which Dimensions DBA "commands" are submitted to the operating system for execution depends on the interface from which they are entered:

■ If the "command" is entered as a `CMDCLIENT` DBA command in the MS-DOS CLI, then the command will be run immediately as a background task (the user will *not* be queried as to whether the task should be run now or sometime in the future). When it completes a mail message containing a log of the command's execution will be sent to the user; this mail message should be examined to determine whether the job was successful.

An exception to the above is *CPAS* (Change Dimensions Database password), which runs in the foreground.

It should be remembered that *CMDCLIENT* "talks" to the Dimensions server specified in the login of the active associated PC Client. For Windows 98, a "remote" login *must* always be specified; therefore the following is delivered to the Dimensions server using the *username specified in the PC Client remote login procedure*:

- The mail message described above.
- Any listings or logs generated by the DBA commands (these will be deposited in the home directory of the username being used on the Dimensions server).

■ If the "command" is entered as a Dimensions DBA command via PC Client's *Run* CLI, then the command will be run immediately as a background task (the user will *not* be queried as to whether the task should be run now or sometime in the future). The execution behavior for a PC Client regarding mail and output listings or logs is the same as that described for *CMDCLIENT* above.

■ If the "command" is entered via the Process Modeler's GUI-based forms, then the "command" will be run immediately as a synchronous task. The Process Modeler's functions applicable to DBA operations are *"Export Process Model"*, *"Create New Base Database"* and *"User Registration".*

Dimensions XCMD Command Script Execution

A number of DBA commands may be run from a command script using the Dimensions XCMD facility (this is described in the "Command-Line Reference Guide"), invoked from the operating-system prompt, the *CMDCLIENT* prompt or PC Client's Run CLI. For example, when creating a large number of Dimensions users, the DBA can place the *CUSR* (Create Dimensions User Database) commands in a script file and then use *pcms xcmd <script-file>* to execute the listed commands.

3 Oracle Maintenance Procedures

In this Chapter

Introduction

NOTE Please refer to the note in "Backup and Recovery" on page 122 regarding MERANT's prerequisites for assisting you in recovering from a database failure.

The Oracle Maintenance Procedures in Dimensions provide facilities to monitor and allocate additional space to your RDBMS database. To take and restore RDBMS database backups, the Oracle export (*EXP*) and import (*IMP*) utilities should be directly used. These utilities are described in "EXP – Oracle Export Utility" on page 78 and "IMP – Oracle Import Utility" on page 80 respectively.

Within the Process Modeler, the function *Export a Process Model* is also available to enable you to export the process model for a specified product into a file. That file may then either be:

- specified at the time of base database creation with the view of importing its process data into the product *$GENERIC* on the newly created base database; or
- used at the time of installation to load the process data into the initial base database created by the installation procedure.

The use of the above Process Modeler function is also described in "Export Process Model to a File" on page 87.

The Oracle Maintenance Procedures of particular concern to Dimensions are normally accessed via either:

- Dimensions commands; or
- Oracle Server Manager command-line utilities like *svrmgrl*.

This chapter also describes how to change the Oracle *SYSTEM* and *SYS* passwords from those set during installation of the Oracle

runtime. This will also be required following the import of an entire system database (see page 82) when these passwords also need to be reset to their default values.

Dimensions provides the following ORACLE Maintenance functions:

- **SPAC–Display RDBMS Database Space Usage**

  ```
  SPAC <qualifiers>
  ```

- **INCR–Increase the RDBMS Database Space**

  ```
  INCR <qualifiers>
  ```

SPAC – Display RDBMS Database Space Usage

Displaying Space Usage

This Dimensions command allows the DBA to display information about database space usage. The syntax is as follows:

```
SPAC [<tablespace>] [/LIST=<listfile>] [/PRINT]
```

where

`<tablespace>`	is the name of the Dimensions tablespace to report on. If omitted, all tablespaces that contain Dimensions data are included in the report.
`<listfile>`	is the file into which the output will be written. If omitted, it will default to *`pcms_space_usage.out`* in the current working directory.

If the */PRINT* qualifier is used, the space information will be sent to the tool specified by the *PCMS_PRINT* symbol.

The information is displayed in the following format:

The first part displays:

- **Tablespace**

  The name of the tablespace.

- **Space Allocated**

  The total space (in Kilobytes) that has been allocated to the RDBMS tablespace. This space will include the initial RDBMS file plus any others subsequently added to the RDBMS tablespace using the function "INCR – Increase the RDBMS Database Space" on page 62.

- **Space Used**

  The total space (in Kilobytes) that has been either used or reserved for use by the RDBMS i.e. space that is not available.

The second part displays information about the space that is still free within the RDBMS database:

- **Number of Fragments**

  The total number of free fragments of space left within the tablespace.

- **Average**

  The average size of these free fragments (in Kilobytes).

- **Smallest**

  The size of the smallest free fragment (in Kilobytes)

- **Largest**

  The size of the largest free fragment (in Kilobytes)

- **Total Free Space**

  The total amount of free space (in Kilobytes)

To successfully create new databases and enable existing ones to grow, enough contiguous blocks must be available. Fewer and larger fragments are preferable to many smaller ones.

Managing Database Space Usage

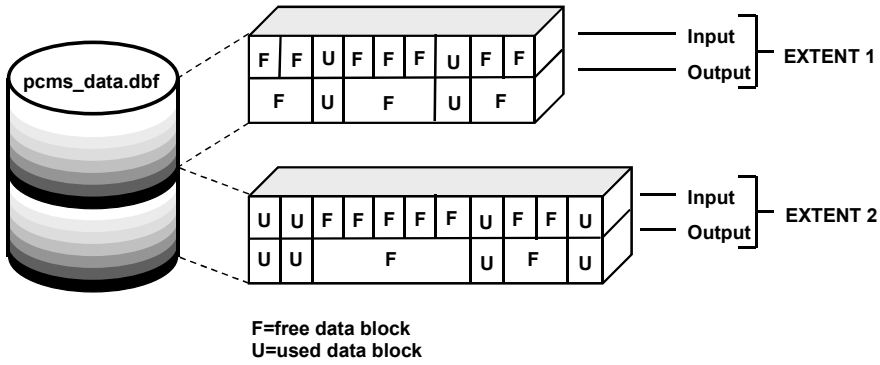

Figure 3-1. Managing Database Space Usage

Within an Oracle database, space is managed using extents that are made up of a specific number of contiguous data blocks. When data is written to the database, the next free extent that is closest in size to the required extent is used; thus a larger free extent can be fragmented, or smaller contiguous free extents can be merged into a larger extent, Figure 3-1. However, this process of continuous allocation and deallocation of free space fragments your database and makes the allocation of larger extents more difficult, which in turn affects the performance of your applications. To monitor the degree of database fragmentation you are recommended to run the *SPAC* utility

described above on a regular basis. If the degree of fragmentation gets too high, then you should consider reclaiming your fragmented space, as described next.

Reclaiming Fragmented Space

When space in either the RDBMS or any of its tablespaces becomes too fragmented, then the DBA should undertake the following activities to reorganize the available free space into contiguous blocks:

- **Reorganizing Available Free Space for a Specific Tablespace**

  Within the context of a single tablespace it is possible to reorganize free space into larger contiguous extents through the use of the Oracle *ALTER TABLESPACE* command. This command can be run through the Oracle Server Manager utility as follows:

  ```
  % svrmgrl
  SVRMGR> connect internal/<sys_password>[1];
  SVRMGR> alter tablespace <tablespace_name> coalesce;
  SVRMGR> exit;
  ```

  where <*tablespace_name*> is the name of the tablespace to process.

- **Reorganizing Available Free Space on a RDBMS Database**

  If you wish to defragment your entire RDBMS database, then you will need to do the following:

  - Shut down Oracle and Dimensions, and restart Oracle in restricted mode (see "Starting and Stopping Oracle" on page 133).

1. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

- Take an operating-system backup of all the database files (normally resident in the UNIX *$ORACLE_HOME/dbs*, or Windows NT/2000 *orant\database* directory).
- Export the entire RDBMS database using the Oracle *EXP* utility (see "EXP – Oracle Export Utility" on page 78 and "Full System Exports" on page 128), or just the users of the tablespace(s) that are badly fragmented. Ensure that when exporting you specify the *COMPRESS=Y* parameter to *EXP*.
- Shut down Oracle (see "Starting and Stopping Oracle" on page 133).
- Re-initialize the database in the manner described in "Importing the Entire System Database" on page 82. Alternatively, offline the named tablespace, and recreate it.
- Import the full RDBMS *SYSTEM* export taken earlier using the Oracle *IMP* utility, or import the export of the tablespace users.
- Finally restart Oracle and Dimensions (see "Starting and Stopping Oracle" on page 133).

This procedure may take some time, but is strongly recommended when the RDBMS database becomes extremely fragmented and performance is impaired.

INCR – Increase the RDBMS Database Space

Database Storage Scheme

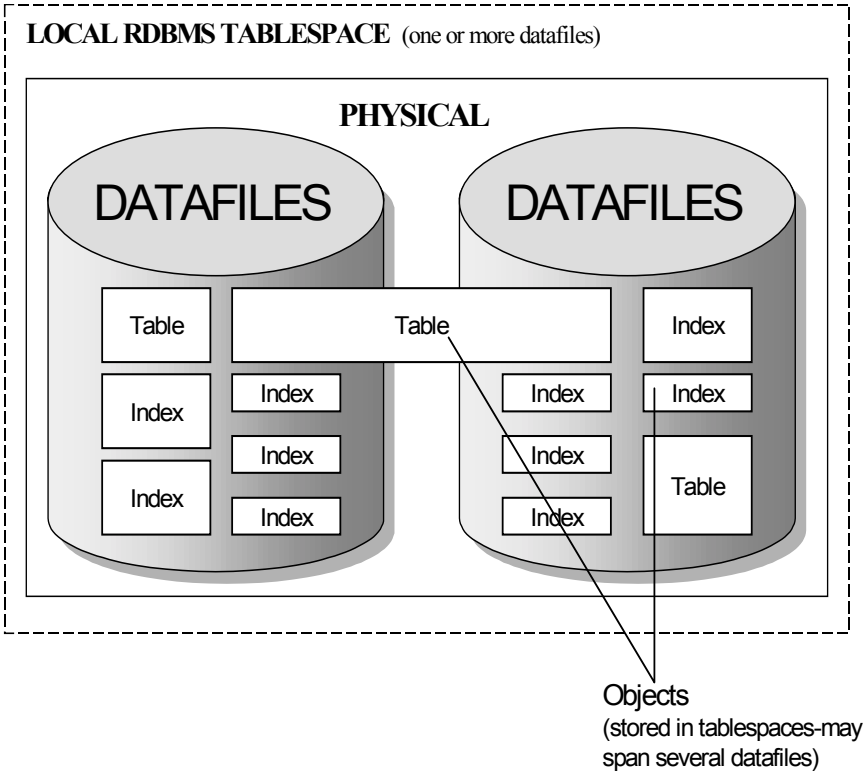

Figure 3-2. Logical and Physical Tablespace Relationship

At the physical level the storage of the database consists of a number of operating system files, each of which is allocated a fixed amount of space.

At the logical level, tables are created in a particular tablespace.

The link between the logical and physical levels is established by the allocation of each operating-system file to one particular tablespace as shown in Figure 3-2 on page 62.

Following a default installation of Dimensions these datafile are located in the following directories:

- *$ORACLE_HOME/dbs* in the UNIX.
- *orant\database* for Windows NT/2000.

You can improve performance on your database by relocating these files across different disks.

Managing Datafiles

When a tablespace becomes full, i.e. the space allocated in the database file is used up, then a message of the following form will be displayed:

```
ORA-01547: failed to allocate extent of size <num> in
   tablespace <name>
```

If this occurs, then additional space can be made available in any of the following ways:

- Recovering internal space previously occupied by data that has been deleted from the database.

  Deleted space is normally reserved for expansion in the tables to which it was initially allocated. It is not available for general use within the database. To reorganize the database and recover this space the DBA must *export*, re-initialise and *import* the entire database (using the Oracle *EXP* and *IMP* utilities).

- Allocating additional database space using the Dimensions *INCR* operation.

Additional space is allocated to a tablespace by providing a new operating-system file for use by that tablespace as a new datafile. The new file will be created and initialized by the RDBMS according to the size as specified by the DBA. These additional operating-system files may, of course, be placed on other disks on the Server machine; however, they must *not* be created on NFS mounted disks on UNIX.

Increasing the Database Space

This Dimensions command allows the DBA to allocate more space to the RDBMS database.

The command syntax is as follows:

```
INCR <tablespace> /FILENAME="<db_file>"[2] /SIZE=<bytes>
```

where

`<tablespace>`	The tablespace to be increased.
`<db_file>`	The name of the file that will hold the additional space specified. If you do not specify a full pathname the file will be created in the current directory. If the file already exists and is not currently used by the RDBMS as a database datafile it will be overwritten. The character '?' may be used to denote *$ORACLE_HOME* on UNIX platforms.
`<bytes>`	The number of bytes to increase the Dimensions tablespace by. It may also be followed by an 'M' or 'K' to indicate Megabytes or Kilobytes

2. Note that the file name *<db_file>* should be enclosed in double-quotes.

Reminder

The additional operating-system file is not constrained to reside in any specific directory or disk (except NFS mounted file systems on UNIX). It is, however, customary to place it:

- For UNIX: in the *$ORACLE_HOME/dbs* directory.
- For Windows NT/2000: in the *orant\database* directory.

However, it should not be created in any other subdirectory in the Oracle directory structure, lest it be deleted during upgrades to the Oracle software.

Notes

The specified disk space must be available on the relevant disk and be within the space-quota of the DBA-account; new space can be allocated on any disk or directory to which the DBA has write access.

This operation must be carried out while both the RDBMS and Dimensions are running. The new space is made available to the RDBMS tablespace concerned immediately.

It is suggested that, if possible, the DBA keeps the additional data files created via this command in the same directory as the other data files to stop the scattering of database files. It is also suggested that the filename used for *<db_file>* should reflect the name of its tablespace, the value of *ORACLE_SID* for the relevant database instance, and the fact that it is an extension file. A suitable format for the filename is *<tablespace>_<orasid>_inc<n>.dbs* (for example *pcms_data_dev7_inc2.dbs* for the second increment to the *PCMS_DATA* tablespace on the *dev7* instance).

It is mandatory that database files are *never* created on NFS mounted disks on UNIX.

Setting the Oracle SYSTEM and SYS Passwords

If you wish to change the Oracle *SYSTEM* and *SYS* passwords from those set (or default-accepted) during installation of the Oracle runtime or following the import of an entire system database, set up your Oracle environment to access the appropriate database and enter the following Server Manager commands:

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[3];
SVRMGR> alter user SYSTEM identified by <new_password_1>;
SVRMGR> alter user SYS identified by <new_password_2>;
SVRMGR> exit
```

3. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

4 Dimensions Database Administration Procedures

In this Chapter

Introduction

The Dimensions-supplied database administration procedures allow the DBA to:

- Create and delete Dimensions databases.
- List Dimensions databases (including who has access to them).
- Administrate tablespaces (see Chapter 3, "Oracle Maintenance Procedures" for details).

These procedures are accessed via Dimensions commands, Process Modeler functions, or both. In all cases, execution of these procedures will result in a mail message being sent to the user upon completion.

The associated Oracle maintenance procedures *EXP* and *IMP* enable the taking and restoring of backups. These procedures are accessed via Oracle command-line utilities.

NOTE All Dimensions database administration procedures, apart from Oracle *EXP* and *IMP*, *must* be performed via the Dimensions interfaces. Do *not* attempt to use Oracle-specific commands to perform functions similar to *CRDB*, *DLDB*, *LSDB* or *Export Process Model to a File* with respect to the Dimensions database.

Functions Provided

For brevity, the Dimensions command names are used when referring generically to the associated functions. The following functions are provided:

- **CRDB–Create Dimensions Base Database**

  Dimensions Command `CRDB <qualifiers>`

  Process Modeler `"User Registration"` function

- **DLDB–Delete Dimensions Base Database**

  Dimensions Command `DLDB <qualifiers>`

- **LSDB–List Dimensions (Base and User) Databases**

  Dimensions Command `LSDB <qualifiers>`

  Process Modeler `"Create New Base Database"` function.

  This is restricted to listing user databases of selected product and account type in a *particular* base database–see "LSDB – List Dimensions User Databases" on page 100.

- **EXP–Oracle Export Utility**

  Oracle Command `EXP <qualifier>`

- **IMP–Oracle Import Utility**

  Oracle Command `IMP <qualifier>`

- **Export Process Model to a File**

  Process Modeler `"Export Process Model function"` function.

CRDB – Create Dimensions Base Database

Dimensions Command

Only users who have *$TOOL-MANAGER* privilege can use this function. The syntax is as follows:

```
CRDB <BaseDb> /GENDB=<generic_db> /USER=<user> -
   [/PRODUCT=<product_id>] -
   [/DEF_TAB=<DefaultTablespace>] -
   [/TEMP_TAB=<TemporaryTablespace>] -
   [/[NO]SECURE] –
   [/[NO]PASSWORD_SAVE] -
   [/ATTRIBUTES=(group_id=<group_id>,site=<site>,-
   Dept=<dept>,full_name=<fullname>,-
   phone=<phoneNo>)]
```

where

`<BaseDb>`	The database name for the new base database that is to be created, e.g. *PROJ_A*. Note that the name of the new base database should be alphanumeric and that the first character must be alphabetic – it is not case-sensitive. For example, *PROJ1* is valid but not *1PROJ*.
`<generic_db>`	The name of an existing base database from which to copy the Generic Process Model, e.g. *PCMS*. Note that the process model information is represented in the product *$GENERIC*.

<product_id>	The name of a product defined in the existing base database given by *<generic_db>* whose process model is to be used to set up the Generic Process Model in the new Dimensions base database. If this is not specified, then the process model for the *$GENERIC* product is used by default.
<DefaultTableSpace>	The default tablespace for the database objects: If this is not specified, objects default to the default tablespace used by *<generic_db>*. The installation default for this is *PCMS_DATA*.
<TemporaryTableSpace>	The temporary tablespace for temporary segments: If this is not specified, temporary segments default to the default tablespace used by *<generic_db>*. The installation default for this is *PCMS_TEMP*.
<user>	The login id of the user who will act as Tool Manager for this Dimensions base database. This user also has exclusive DBA privilege for the new base database (enabling *CUSR* operations).
<site>	The location or site of the designated Tool Manager, e.g. *St. Albans*
<group_id>	The user's group, e.g. *Testing*
<dept>	The department of the designated Tool Manager, e.g. *Development*.
<fullname>	The full name of the Tool Manager, e.g. *Joe Dunlop*.
<PhoneNo>	The telephone number of the Tool Manager, e.g. *01727 812812*

Experienced Oracle users may note that Dimensions DBA commands will by default support the use of RDBMS tablespaces other than the *SYSTEM* tablespace when an Oracle user is created. By default the *PCMS_DATA* tablespace is used.

The length of each item of information regarding the new base database is limited, and any excess will be truncated. In particular, the database names, passwords and login-id are limited to 25 characters, and the department field to 10 characters. The Dimensions *DBA PROF* function (see "PROF – Display User's Profile Information" on page 101) enables the DBA to post-edit these fields.

This command will create a new base database called *<BaseDb>*, and an associated user database called *<BaseDb>_tool*. The new Tool Manager can connect to the base database via the *<BaseDb>_tool* user database. Password assignment is as follows:

Security Parameter	Password for Base Database <BaseDb>	Password for User Database *<BaseDb>_tool* [a]
None	Non-secure. Plain text password *<BaseDb>*	Secure. Encrypted password.
/NOSECURE	Non-secure. Plain text password *<BaseDb>*	Non-secure. Plain text password *<BaseDb>_tool/ <BaseDb>_tool*
/SECURE	Secure. Encrypted password.	Secure. Encrypted password.

a. See "Security Considerations" on page 38.

Certain Dimensions PC Client tools[1] have a "remote automatic login" facility (see, for example, the *PC Client User's Guide*). To be able to utilize this facility, the user's operating system login password on the remote Dimensions server (UNIX or Windows NT/2000) must be saved locally on the client PC. To enable the remote automatic login for the Tool Manager, the */PASSWORD_SAVE* qualifier must be specified. By default, this facility is not enabled and can also be explicitly disabled by specifying the */NOPASSWORD_SAVE* qualifier. Please refer to "CUSR – Create Dimensions User Database" on page 92 for further details of client automatic remote logins.

On completion of the command, the DBA will receive a mail message of the Dimensions Batch Log, and the Tool Manager will be sent an advisory mail message describing the details on the new base database and the changes to the appropriate environment for running Dimensions.

Process Modeler

Only users who have *$TOOL-MANAGER* privilege can use this function. The function is accessed as follows:

```
Create New Base Database
```

This function enables a new Dimensions base database to be created either by basing it on an existing Dimensions base database or by importing a Dimensions process model export file. The created base database will be of type *SECURE* and will default to *PCMS_DATA* and *PCMS_TEMP* for the default and temporary table spaces respectively – you *cannot* override these assignments within the Process Modeler. The parameters corresponding to the */ATTRIBUTES* qualifier in command mode are entered as a *post* base-database action using the Process Modeler function:

```
User Registration \(Select Account Type) \Edit
```

1. Dimensions Make, PC Client, PVCS Project Merge and SCC Interface.

Refer to the description of the *CRDB* command and the related document *Process Modeling User's Guide* for more detailed information.

DLDB – Delete Dimensions Base Database

Dimensions Command

This option allows a DBA with DBA privilege on a base database to delete that base database. To be able to use this command, you must issue it while connected to *another* base database i.e. it is *not possible* to delete a base database while any users are connected to it.

All information contained in the base database will be deleted and all the Dimensions users who were defined to use it will also be dropped. Note, however, that the product item libraries (which are operating-system directories) associated with the base database are *not* deleted.

CAUTION! The DBA must first check that the contents of the base database are no longer needed as the deletion operation cannot be 'undone' later on (except by resorting to a backup copy of the database files or a full export of the RDBMS *SYSTEM* taken before deletion). The *DLDB* operation *must* be run when no other users are accessing the base database to be deleted. If users are accessing the base database then the deletion will not occur, although some user databases may be dropped.

The *DLDB* function does check if there are any such users prior to its running, but this does not prevent users attempting to connect to the base database or to any associated user database during the operation of the *DLDB* function.

The syntax is as follows:

```
DLDB <BaseDb>
```

where.

`<BaseDb>` is the base database name

Notes

1 *DLDB* deletes the base database and any database objects owned by that base database.

2 *DLDB* can take a long time to complete, especially if there are a lot of user databases.

3 The associated Oracle user *OPS$<login id>* (where *<login id>* is the operating system name of the DBA e.g. *OPS$PCMS*) may no longer be a DBA for any base database. Such users must be removed using *sqlplus*, for example:

```
sqlplus system/<password>
SQL> drop user OPS$PCMS;
```

Process Modeler Not supported.

LSDB – List Dimensions Database (Base and User)

Dimensions Command This option enables the DBA to display details on base and user databases that have been created, and users authorized to use them by the CUSR or UREG commands. The syntax is as follows:

```
LSDB [<BaseDb>] [/LIST=<listfile>] [/PRINT]
```

where

`<BaseDb>` is the name of the base database to report on. If this parameter is omitted then all the base databases in the RDBMS will be included in the report.

`<listfile>` is the name of the file into which the output will be written. If omitted it will default to *`pcms_list_dbs.out`* in the current directory

If the */PRINT* qualifier is used, the output file will be sent to the tool specified by the *PCMS_PRINT* symbol.

An example output file generated by this operation is shown below.

In this example report, user DIMTEST2 for product FS was created using UREG; whereas, all others users where created using CUSR.

```
List of Base and related User databases on 17:49:19 Friday 18 January 2002
  Dimensions Databases  _________  USER Databases
  FS                    <-------   DIMTEST
                        <-------   FS
                        <-------   FS_TOOL
                        <-------   FS_USER1
  INTERMEDIATE          <-------   INTERMEDIATE
                        <-------   INTERMEDIATE_REPORT
                        <-------   INTERMEDIATE_TOOL
                        <-------   INTERMEDIATE_USER2
                        <-------   INTERMEDIATE_USER3
                        <-------   INTERMEDIATE_USER4
                        <-------   INTERMEDIATE_USER5
                        <-------   INTERMEDIATE_USER6
```

```
List of products contained in FS
  Dimensions Databases          Product id
  FS                            $GENERIC
                                FS

List of users for FS
  Dimensions User               USER/Registration Database
  CTL                           FS_TOOL
  DIMTEST                       DIMTEST
  DIMTEST2                      FS
  USER1                         FS_USR1

List of products contained in INTERMEDIATE
  Dimensions Databases          Product id
  INTERMEDIATE                  $GENERIC
                                PAYROLL

List of users for INTERMEDIATE
  Dimensions User               USER/Registration Database
  *                             INTERMEDIATE_REPORT
  CTL                           INTERMEDIATE_TOOL
  USER2                         INTERMEDIATE_USER2
  USER3                         INTERMEDIATE_USER3
  USER4                         INTERMEDIATE_USER4
  USER5                         INTERMEDIATE_USER5
  USER6                         INTERMEDIATE_USER6
```

Process Modeler

`"User Registration \ (Select Account Type)"` function. This is restricted to listing Dimensions user databases – see page 100.

EXP – Oracle Export Utility

Oracle Command

The Oracle Export utility allows the DBA to copy the contents of the RDBMS database into a file for either archiving or restructuring the database space (i.e. by using the Oracle Import Utility).

The Oracle Export utility (*EXP*) is available with the following options:

USERID	The RDBMS database name and password to export.
BUFFER	The operating-system dependent size in bytes of the data buffer used to fetch data rows.
FILE	The output file created as a result of the export process (default *EXPDAT.DMP*).
GRANTS	A flag to indicate that grants should be exported (default *Y*).
INDEXES	A flag to indicate that indexes should be exported (default *Y*).
LOG	Log file for informational and error messages from the *EXP* command.
FULL	A flag to indicate that the entire database be exported (default *N*).
COMPRESS	A flag to indicate that database extents should be compressed into one (default *Y*).
CONSTRAINTS	A flag to indicate that constraints should be exported (default *Y*).
OWNER	A list of usernames whose objects should be exported.
TABLES	A list of table names to be exported.

ROWS	A flag to indicate whether or not rows of table data should be exported (default *Y*).
HELP	Enter *Y* for help.

The following are typical examples of the *EXP* command:

- To export an entire RDBMS use the following

```
exp userid=system/manager buffer=20000 file=system.exp
   full=y compress=y
```

  See also "Full System Exports" on page 128.

- To export a Dimensions base database use the following:

```
exp userid=pcms/pcms file=pcms.exp
```

- To export a selection of Dimensions base database tables:

```
exp userid=pcms/pcms file=pcms_tables.exp
   tables=(part_spec_catalogue,item_sets)
```

Cautions

1. It is recommended that, for an entire RDBMS database export, the export operation should only be run when no other users are accessing the affected database. This can be achieved by shutting down the RDBMS and restarting it in restricted mode as described in "Starting and Stopping Oracle" on page 133.
2. Prior to running *EXP* it is recommended that you ensure that the environment variable *NLS_LANG* is unset. This will ensure your database is exported in the correct National Language character set.

IMP – Oracle Import Utility

Oracle Command

The Oracle Import utility (*IMP*) enables the DBA to copy the contents of an export file back into the database.

IMP is available with the following options:

USERID	The RDBMS database user name and the password into which to import the data from the export file.
BUFFER	The operating-system dependent size in bytes of the data buffer used to fetch data rows.
FILE	The output file created as a result of the export process (default *EXPDAT.DMP*).
SHOW	Print to standard output what the IMP command would do (default *N*).
IGNORE	Specifies whether or not import will ignore creation errors (default *N*).
LOG	Log file into which to capture the output from the *IMP* command.
TABLES	A list of table names to import (* for all tables).
FULL	A flag to indicate that the full database details specified by the *USERID* be imported (default *N*).
GRANTS	A flag to indicate that grants should be imported (default *Y*).
INDEXES	A flag to indicate that indexes should be imported (default *Y*).
ROWS	A flag to indicate whether rows of table data should be imported (default *Y*).
FROMUSER	List of owner usernames from which table data will be imported.
TOUSER	List of owner usernames into which table data will be imported.

HELP	Enter *Y* for help.
COMMIT	Commit data during the import process (default *N*). Use this option for large database imports.
RECALCULATE_ STATISTICS	A flag to force the import to recalculate the statistics used by the cost-based optimizer (default *N*). Whenever you do an import it is recommended that you set this flag to *Y*.

The following is a typical example of the *IMP* command:

- To import an entire RDBMS use the following (see, also, "Importing the Entire System Database" on page 82)

```
imp userid=system/manager buffer=20000 file=system.exp
   full=y ignore=y commit=y recalculate_statistics=y
```

Cautions

1. *IMP* should *only* be used to import an export file into an *empty* database. *Do not* import an export file into an existing *non-empty* database as this may cause *duplication* and *corruption* of data.

2. Following the import of Oracle data into a Dimensions database schema, each sequence generator used in the database must be re-initialized to a value in excess of any value used in the base database. A Dimensions script is provided to accomplish this task – see "Set Dimensions Sequence Values" on page 106.

3. For entire RDBMS database imports, the import operation should only be run when there are no other users accessing the target database. This can be achieved by shutting down the RDBMS and then restarting it in restricted mode as described in "Starting Oracle in Restricted DBA Mode" on page 136. After the import is complete it should be shut down and restarted in normal (or shared, if appropriate) mode. For individual database imports the import operation

may be performed while the database is running in normal mode, but ensure that no users are accessing the database into which information is being imported.

Importing the Entire System Database

When the DBA specifies the entire RDBMS database (i.e. *USERID=system/<password>)*, all databases found in the export file will be imported. This operation requires the re-creation and initialization of the entire RDBMS database before the import operation is started.

Before re-creating the database in the manner described below, it is *strongly* recommended that the RDBMS database files are backed up (see "Performing a Full Backup" on page 124) and the entire RDBMS database exported (see "Full System Exports" on page 128). As discussed in "Oracle Version 8i Runtime" on page 146, the RDBMS database files corresponding to an Oracle Instance typically include four or more operating-system data files, two redo log files and one or more control files. At installation time these files are placed in the UNIX *$ORACLE_HOME/dbs* or Windows NT/2000 *orant\database* directory, or elsewhere as specified.

NOTE A tablespace may have more than one data file depending on whether or not the database space has been increased using the Dimensions *INCR* command (or a new tablespace has been added using Oracle's command-line utilities).

Having successfully exported the entire RDBMS database, shut Oracle down before re-creating the database (see "Stopping Oracle During DBA Operations" on page 135). Make a note of the current operating-system data files and redo log files, the directories in which they reside and their size. You can do this by using the following commands:

```
% sqlplus system/<password>
SQL> spool <filename>
SQL> select * from sys.dba_data_files;
SQL> select * from v$logfile;
SQL> select * from v$controlfile;
SQL> spool off
SQL> exit;
```

This command will provide a list of all the files that the Oracle instance uses, their sizes, and where they are located.

Determine whether you wish to re-create the data file(s) and the redo log files with larger sizes. It is recommended that the size of the log files be somewhere in the range 2–4 Mbytes for a database of size up to 100 Mbytes. The following guidelines will help you determine a suitable size for the new database file:

- As a provisional estimate, 1,500 items under the control of Dimensions with an average of five revisions may generate a database requirement of 50 Mbytes.
- Each base database will require a minimum of 50 Mbytes in the *PCMS_DATA* tablespace
- Each user database will require a minimum of 0.5 Mbytes in the *PCMS_DATA* tablespace.

NOTE Please refer to the *README* file for any later recommendations.

To re-create the entire RDBMS database, run the command file or installer option as described next. Ensure first, however, that you read and act upon the following precautionary caution and notes:

CAUTION! Prior to proceeding with these commands you *must* ensure that you have a valid backup/export of your database.

Notes

1 A full backup of all the operating-system database files and redo log files should be made before the import operation is started.

2 The DBA should wait until the import has been completed successfully before accessing or letting any other users access either Dimensions or the RDBMS. At this time the database will need to be restarted in normal mode.

- For UNIX:

  ```
  % sh $ORACLE_HOME/dbs/create_<db_name>.sh
  ```

  This script must be run from a privileged operating-system user account (e.g. *root*), and you must delete the database files (*.dbf*) referenced before running the script.

- For Windows NT/2000:

  Having uninstalled the old database, execute the installer and select *"Create Dimensions Oracle Instance"*.

This file is created at the time of the Dimensions installation with a filename derived from the database name. This command file uses the Oracle Server Manager Line Mode (*SVRMGRL*) utility to re-create and initialize the RDBMS database exactly as it was at the time of installation.

The command file above will create a log file:

- For UNIX:

  ```
  /tmp/create_<db_name>.log
  ```

- For Windows NT/2000:

  ```
  <ORACLE_Home>\Database\*.log
  ```

which should be checked for any errors that may have occurred during the creation process.

After the database is successfully re-created, restart the database in restricted mode (see "Starting Oracle in Restricted DBA Mode" on page 136) and import the entire system export. (Note the passwords for the RDBMS users *SYSTEM* and *SYS* will have reverted to their defaults.) However, if other products using the Oracle database make use of tablespaces other than *SYSTEM*, then it is important that additional rollback segments are created for each tablespace (using the appropriate Oracle commands). On import, the additional tablespaces will be created and the RDBMS *SYSTEM* password restored. However the password for *SYS* will not be restored and should be set manually.

The setting and resetting of the Oracle *SYSTEM* or *SYS* passwords are described on page 66.

Importing a Specific Dimensions Base Database

This operation recreates the specified Dimensions database and loads the contents of the export file enabling the DBA to recreate the contents of a database that may have been previously either deleted or corrupted.

The following should be noted and/or acted upon:

- A full backup of all the operating-system database files must be made before the import operation is started.
- The contents of the specified base database must first be deleted to avoid clashes (e.g. duplicate information) with the imported data.
- The DBA must ensure that the correct database and export file names are specified for input to ensure the consistency of the contents of the database.

- This operation can only be used for importing base databases into the *same* RDBMS from which the export was made.
- Do *not* import into a database created using the Dimensions *CRDB* operation as this will have Dimensions data installed. Instead you can create the database to receive the import using the following SQL*Plus commands:
  - Firstly, check to see if the user already exists:

    ```
    sqlplus system/<system password>
    SQL> select * from all_users
    2> where username='<username>';
    ```

  - If the user exists:

    ```
    SQL> drop user <username> cascade;
    ```

    Before running the above command check that the user to be deleted is the correct one, that any backups of the user are made (via ORACLE *EXP*), and finally that no one is connected to the ORACLE RDBMS as that user.
  - Then create the database:

    ```
    SQL> create user <username> identified by <password>
    2> default tablespace PCMS_DATA
    3> temporary tablespace PCMS_TEMP
    4> quota unlimited on PCMS_DATA
    5> quota unlimited on PCMS_TEMP;
    SQL> grant connect, resource to <username>;
    SQL> exit
    ```

  - Finally, import the export file into the RDBMS database.

Reducing Database Fragmentation

A RDBMS database may over time become increasingly fragmented as Dimensions users are created and dropped and Dimensions operations are performed. This fragmentation may begin to have a harmful effect on the performance of Dimensions if left unchecked. The DBA can check on the amount

of fragmentation that is present in the database by running the Dimensions *SPAC* report (see page 57). This report details the number of fragments that are present in a database. If the number under the column "Number of Fragments" reads at or above 50 then the DBA should consider performing the operations specified in "Importing the Entire System Database" on page 82, once a *SYSTEM* export of the RDBMS has been performed.

Alternatively, performance may be significantly improved by dropping and recreating the indexes at regular intervals, as these too may become fragmented. An SQL script in *PCMS_DBASE* is provided for this purpose – *create_pcms_index_script.sql*. This will produce a script to drop and disable constraints and indexes, and recreate them with optimum storage definitions. The resulting script may be run overnight or individual indexes can be selected for recreation.

Export Process Model to a File

Process Modeler

The function is accessed as follows:

```
Export Process Model
```

This function enables a process model, defined for a particular product, to be exported into an (operating-system) export file. An associated log file is generated. See "Create Generic Export File" on page 107.

The export file generated can then be specified as an input:

- When creating a Dimensions base database. Here the process data in the export file is imported into the product *$GENERIC* of the newly created base database. See "CRDB – Create Dimensions Base Database" on page 70.

- When installing Dimensions. Here the process data in the export file is loaded into the initial base database created by the install program. See the relevant Dimensions Installation Guide (separate document for each operating system).

Refer to the description of the *CRDB* command (see page 70) and the related document *Process Modeling User's Guide* for more detailed information.

Error Recovery

- The creation of a new database may fail if the associated operating-system disk space it requires is either more than that is physically available to the DBA or for some reason is not accessible by the DBA. To recover from this the DBA must either obtain the required disk space (consult the System Manager) or reduce the current database space request. To reduce the space you must change the create command file before resubmitting it for execution.
- Database creation may also fail if the `ulimit` on System V UNIX System is too low. The operating-system administrator will need to increase the value of this parameter if this is the case.

5 Dimensions User Administration Procedures

In this Chapter

Introduction

The Dimensions user administration procedures allow the DBA to regulate which users have access to Dimensions. These procedures are provided as Dimensions commands, Process Modeler functions or both.

However, for Dimensions 7.1 the registration of users has been modified making it simpler. Please refer to "Modified User Registration for Dimensions 7.1" on page 102.

NOTE All Dimensions user administration procedures *must* be performed via the Dimensions interfaces. Do *not* attempt to use Oracle-specific utilities to perform functions similar to *CUSR*, *DUSR*, *CPAS*, *LSDB*, *PR0F* or *CTM* operations with respect to the Dimensions database.

For brevity, the command names are used when referring generically to the associated functions. The following functions are provided.

- **CUSR – Create Dimensions User Database**

  Dimensions Command `CUSR <qualifiers>`

  Process Modeler `"User Registration \ (Select Account Type) \ Add"` function.

  NOTE: If you select 'Normal', UREG will be invoked.

- **DUSR – Delete Dimensions User Database**

  Dimensions Command `DUSR <qualifiers>`

  Process Modeler `"User Registration \ (Select Account Type) \ Delete"` function.

  NOTE: If you select 'Normal', XREG will be invoked.

- **CPAS – Change Dimensions Database Password**

  Dimensions Command `CPAS <qualifiers>`

- **LSDB – List Dimensions (Base and User) Databases**

  Dimensions Command `LSDB <qualifiers>`

  This is described in "LSDB – List Dimensions Database (Base and User)" on page 75.

  Process Modeler `"User Registration \ (Select Account Type)"` function.

  This is restricted to listing user databases (of the selected product and account type) in the base database for which the Process Modeler has been launched i.e. you cannot respecify the base database here.

- **PROF – Maintain User Profile Information**

  Process Modeler `"User Registration \ (Select Account Type) \ Edit"` function.

- **CTM – Change Tool Manager**

  Process Modeler `"User Registration \ (Tool Manager) \ Add"` function.

NOTE This command can only be run on the Dimensions server.

Also described within this chapter are additional Dimensions scripts used for specialized purposes (see "Additional Utilities" on page 106).

CUSR – Create Dimensions User Database

NOTE Post Dimensions 7.1, apart from proxy and report user creation, UREG (page 105) provides a better way of managing user access.

Dimensions Command

This command will create a Dimensions user. A DBA may only create user databases in base databases for which that user has DBA privilege (see also "CTM – Change Tool Manager" on page 101).

The syntax is as follows:

```
CUSR <UserDb> /BASEDB=<base_db> -
   /USER=<user> | /PROXY [/USER=<user>] | /REPORT -
   [/WORKSET=<workset_spec>] -
   [/DEF_TAB=<DefaultTablespace>] -
   [/TEMP_TAB=<TemporaryTablespace>] -
   [/[NO]SECURE] -
   [/[NO]PASSWORD_SAVE] -
   [/ATTRIBUTES=(site=<site>,group_id=<group_id>, -
   Dept=<dept>,Full_name=<fullname>,phone_no=<PhoneNo>)]
```

where

`<UserDb>`	The name of a new Dimensions user database, e.g. *USER_A*.
`<base_db>`	The name of an existing Dimensions base database the new user will be allowed to access, e.g. *PROJ1*.
`<user>`	The login id of the new Dimensions user, e.g. *usera*.
`<workset_spec>`	The specification (in the form *<product_id>:<workset_name>*) of the user's initial default workset. If this is not specified, no initial default workset is defined for the user.

<DefaultTableSpace>	The default tablespace for the user's database objects. If this is not specified, objects default to the tablespace used by *<base_db>*. The installation default for this is *PCMS_DATA*.
<TemporaryTableSpace>	The temporary tablespace for the user's temporary segments. If this is not specified, temporary segments default to the tablespace used by *<base_db>*. The installation default for this is *PCMS_TEMP.*
<site>	The location or site of the new Dimensions user, e.g. *St. Albans*
<group_id>	The group id of the new Dimensions user, e.g. *Testing*.
<dept>	The department of the new Dimensions user, e.g. *Manufacture*.
<fullname>	The full User name of the new Dimensions user, e.g. *A. N. User*
<PhoneNo>	The new Dimensions User's telephone number, e.g. *01727 812812*

This command normally creates a user database whose type is determined by the */USER*, */PROXY* or */REPORT* qualifiers (only one qualifier allowed). For */USER* a standard user database is created, and this is set up for use from accounts logging on to the operating system as <user>. For */PROXY* a guest user database is created, and for */REPORT* a reporter account is created. Both of these user databases may be accessed from any operating-system account.

If */USER* and */PROXY* are both specified, then the user database must have already been created as a */PROXY* user database. This form of command is used to provide details on a user before they make unregistered use of the user database (see "Unregistered User Database Account" on page 39).

By default, both standard and guest user databases are created with encrypted passwords, but a reporter user database is created with a password of *<UserDb>*. If */SECURE* is specified, then all types of user database will be given encrypted passwords. If */NOSECURE* is specified, then all user databases will have plain text passwords initially set to *<UserDb>*.

Certain Dimensions PC Client tools[1] have a "remote automatic login" facility (see, for example, the *PC Client User's Guide*). To be able to utilize this facility, the user's operating system login password on the remote Dimensions server (UNIX or Windows NT/2000) must be saved locally on the client PC. When you create a Dimensions user

- This facility is permitted by specifying the */PASSWORD_SAVE* qualifier. Subsequently, the first time a client tool is invoked the user will have the option in the Remote Login dialog of storing their remote login password in an encrypted format in the client PC's Windows Registry (under the existing connection specific key with an Access Control List (ACL) applied to it for added security). Further remote logins will than be automatic unless the *CTRL* or *Shift* key is kept depressed during client invocation.
- This facility is deactivated by specifying the */NOPASSWORD_SAVE* qualifier. If neither the */PASSWORD_SAVE* nor the */NOPASSWORD_SAVE* is specified then the latter is the assumed default.

If a Dimensions user is created without activating remote login password saving, this facility can be subsequently activated using the Dimensions *CPAS* command and *vice versa* (see page 98).

If the user subsequently changes their remote login password:

- The next time they try an automatic remote login from one of the specified client tools, they will receive a message

1. Dimensions Make, PC Client, PVCS Project Merge and SCC Interface

informing them that their stored remote login password is incorrect.

- The Remote Login dialog will be displayed prompting them for their (new) remote login password.
- Once connection has been established, the new remote login password will get stored in the Windows Registry as described above.
- From then on, subsequent remote logins will again be automatic.
- All Dimensions DBA commands support the use of RDBMS tablespaces, other than the *SYSTEM* tablespace when an Oracle user is created. On installation, the *PCMS_DATA* tablespace is used by default.
- Definition of a new Dimensions user results in the creation of a personal database for that user. This database contains tables that hold temporary data generated by various query applications that the user may run within Dimensions. It also has "grants" that allow the user to view Dimensions data held within the base database.

Each user database must have a distinct name, *which must also differ from that of any Dimensions base database name*. If only one Dimensions base database is used, then it may be helpful to use the user's operating-system login id as the personal database name. If access to more than one Dimensions base database is required for any *one* user, then a separate Dimensions user database must be created for each base database. The user must set *PCMSDB* (an environment variable in UNIX and Windows NT/2000) to the appropriate user database name (and optionally set a password, if created */NOSECURE*) before running Dimensions to select which Dimensions base database is to be accessed.

The length of each item of information about the new user is limited, and any excess will be truncated. In particular, the

database names, passwords and login-id are limited to 25 characters, and the department field is limited to 10 characters.

An advisory mail message is sent to the user giving details on changes to the appropriate environment for running Dimensions. The Dimensions Batch Log is mailed to the DBA.

Process Modeler

The function is accessed as follows:

```
User Registration | (Select Account Type) | Add
```

This function is only available if the Dimensions user database type *Normal*, *Proxy* or *Report* is selected from the Select Account Type combo box (these type names correspond to the */USER*, */PROXY* and */REPORT* qualifiers of the *CUSR* command). The Dimensions user database will be created without an initial default workset and with the default values of *PCMS_DATA* and *PCMS_TEMP* for the default tablespace and temporary table space respectively. You *cannot* override these assignments within the Process Modeler.

Refer to the description of the *CUSR* command (see page 92) and the related document "Process Modeling User's Guide" for more detailed information.

DUSR – Delete Dimensions User Database

NOTE Post Dimensions 7.1, apart from proxy and report user creation, XREG (page 105) provides a better way of managing user access.

Dimensions Command

User databases should be deleted whenever they no longer need to use Dimensions. Deleting a user reclaims the resources allocated to that user's personal database.

This command enables the DBA for a base database to delete users from it. The syntax is as follows:

```
DUSR <UserDb> [/[NO]KEEP]
```

where

`<UserDb>`	is the name of the Dimensions user database to be deleted, e.g. *USER_A*.
`/[NO]KEEP`	determines whether associated delegations and user role assignments are to be kept or deleted.

CAUTION! For this function to run successfully, it is important to ensure that no other Dimensions or DBA activity, which affects this user database, should occur while it is running. The operation must also not be interrupted or terminated prematurely.

Prior to the *DUSR* operation being started, the command will check to see if any users are currently connected to the user database. The operation will not be attempted if the user database is in active use. However, the *DUSR* operation does not prevent users attempting to connect while the operation is in progress.

Once the *DUSR* operation has completed, an advisory mail message is sent to the user with details on the user database that has been deleted.

DUSR deletes the user database and drops *<UserDb>* as a database user. This means that any other database objects owned by that user are also deleted. *DUSR* retains the corresponding user profile information. However, unless */KEEP* is specified, associated delegations and user role assignments (specified by the Dimensions *DUR* Define User Roles command or Process Modeler *Role Definitions and Assignments* operation) are also deleted.

Process Modeler

This function is accessed as follows:

```
User Registration | (Select Account Type)| Delete
```

This function is only available if the user database type *Normal*, *Proxy*, *Report* or *Tool Manager* is selected from the *Select Account Type* combo box (such a selection is purely made in order to display the appropriate user database name ready for deletion). Type names *Normal*, *Proxy* and *Report* correspond to the */USER*, */PROXY* and */REPORT* qualifiers of the *CUSR* command; whereas the type name *Tool Manager* enables you to demote a Tool Manager back to a normal Dimensions user.

The user database will be deleted using the default */KEEP*. You *cannot* override this assignment from within the Process Modeler.

Upon completion of the delete operation, the deleted user's account profile information is retained within the base database, and such a user becomes *Dormant*. The identity of dormant users can be displayed by selecting type name *Dormant* from the *Select Account Type* combo box. A dormant user's account profile information can also be updated using the associated **Edit** button.

Refer to the description of the *DUSR* command (see page 96), the description of the *CUSR* command (see page 92) and the related document *Process Modeling User's Guide* for more detailed information.

CPAS – Change Dimensions Database Password

Dimensions Command

The password of a Dimensions user may be changed, without affecting the contents of the database. This command is executed by the DBA who is responsible for the base database.

The syntax is as follows:

```
CPAS <dbname> -
/NEW_PASSWORD=<user_passwd> | /SECURE] –
[/[NO]PASSWORD_SAVE]
```

or

```
CPAS <dbname> -
[/NEW_PASSWORD=<user_passwd> | /SECURE] –
/[NO]PASSWORD_SAVE
```

where

`<dbname>`	is the Dimensions user database name whose password is to be changed.
`<user_passwd>`	is the new (plain text) password for the database.

One of the following qualifiers must be specified:

- */NEW_PASSWORD* or */SECURE*. A plain text or encrypted password respectively will be assigned.
- */NOPASSWORD_SAVE* or */PASSWORD_SAVE*. Indicates whether or not a Dimensions user has permission to save their remote login password. Please refer to "CUSR – Create Dimensions User Database" on page 92 for further details of client automatic remote logins.

When the RDBMS password is changed, the user must also change the assignment to the environment variable *PCMSDB*. A mail message is sent to remind the user of the changes necessary for running Dimensions.

For example.

- For UNIX:

  ```
  setenv PCMSDB database_id/new_passwd
  ```

  or

```
setenv PCMSDB database_id[3]
```

- For Windows NT/2000:

```
$ set pcmsdb=database_id/new_passwd[2]
```

or

```
$ set pcmsdb=database_id[3]
```

Process Modeler

Not supported.

LSDB – List Dimensions User Databases

Dimensions Command

This function enables a DBA to:

- Display names of base databases.
- Display names of Dimensions users set up within each base database.
- Display names of products contained in each base database.

This is described in "LSDB – List Dimensions Database (Base and User)" on page 75. Listing Dimensions user databases forms just part of the full listing.

Process Modeler

The function is accessed as follows:

```
User Registration | (Select Account Type)
```

This function is restricted to listing user databases (of the selected product and account type) in the base database for which the Process Modeler has been launched i.e. you cannot respecify the base database "on the fly". Listed are the *PCMSDB*, *`User Id`*, *`Full Name`* and *`Telephone Number`*.

2. Plain password.
3. Encrypted password.

Refer to the related document *Process Modeling User's Guide* for more detailed information.

PROF – Display User's Profile Information

Dimensions Command

Not supported.

Process Modeler

The function is accessed as follows:

```
User Registration | (Select Account Type) | Edit
```

This function enables a DBA to select a displayed Dimensions username (including a *Dormant* username) and edit the following account information: *Full User Name*, *Telephone Number*, *Group Id*, *Site* and *Department*.

Refer to the related document *PC Client User's Guide* for more detailed information.

CTM – Change Tool Manager

Dimensions Command

Not supported.

Process Modeler

The function is accessed as follows:

```
User Registration | (Tool Manager) | Add
```

This function is only available if the user database type *Tool Manager* is selected from the *Select Account Type* combo box. The function enables an existing Tool Manager to add a Dimensions user to the list of Tool Managers for the base database (the base database concerned is the one that was

selected when the Process Modeler was first invoked). This operation should only be attempted when running on the Dimensions server.

Tool Managers are deleted (to be precise, demoted back to normal Dimensions users) by use of the following Process Modeler function:

```
User Registration |(Tool Manager) | Delete
```

Refer to the related document *Process Modeling User's Guide* for more detailed information.

Notes

1 The Dimensions user to be promoted must have been created with an account type of Normal – please refer to "CUSR – Create Dimensions User Database" on page 92.

2 As a DBA, the Tool Manager is empowered to use the *CUSR* command on this base database. You can also use the *CRDB* command to create new base databases based on this base database.

Modified User Registration for Dimensions 7.1

In previous releases of Dimensions, all users were required to use separate Oracle accounts for accessing the Dimensions database. This was required only because of the way temporary tables were handled by Dimensions in the Oracle database. In this release temporary tables are being handled differently and in a way that eliminates the need for a separate Oracle account for each Dimension user. A result of this optimization is that user registration is now considerably faster (a few seconds compared to a few minutes) and also completely eliminates the database

storage overhead previously incurred (about 0.75Mb per user) to create user database accounts.

The following commands are made available to register users in this new way. Once users are registered in this way, they will be able to access Dimensions by specifying the name of the Dimensions base database in the PCMSDB symbol or in the "Database Login Id" field of the connection dialog. A requirement for this feature is that the Tool Manager Oracle account for the base database (<basedb>_tool) must use a "secure" password.

NOTE All Dimensions user administration procedures *must* be performed via the Dimensions interfaces. Do *not* attempt to use Oracle-specific utilities to perform functions similar to *UREG*, *XREG*, *MREG* operations with respect to the Dimensions database.

- **UREG - Register Users**

  (alternative to CUSR)

Dimensions Command	UREG <qualifiers>
Process Modeler	"User Registration | Account Type - Normal | Add" function.

- **XREG - Un-register User**

  (alternative to DUSR).

Dimensions Command	XREG <qualifiers>
Process Modeler	"User Registration | Account Type - Normal | Delete" function..

- **MREG - Migrate User**

  Dimensions Command MREG <qualifiers>

The aim of this modified registration is to implement a simple solution for Dimensions 7.1 that can run alongside the existing scheme and has no impact on existing users, but which also allows new users to be registered without creating a new Oracle user account.

The advantages of this approach are:

- It is much quicker to administer Dimensions users
- There is a saving of 0.75Mb disk space in the Oracle database per user.

The new commands UREG and XREG perform the same operations on metadata as CUSR and DUSR, but will not create or delete any Oracle users.

The new command MREG migrates an existing user by removing the associated Oracle account and updating metadata to allow use of the Tool Manager database account.

These "new style" users will be SECURE i.e. have an encrypted password.

Users registered by UREG will use the Tool Manager account for the base database

Temporary table names are of the form `QXQX_%S`

where

- `%S` is a unique string

UREG – Register Users

Dimensions Command

UREG uses the following parameters in the same way as CUSR:

- <UserId> – the user's login id
- /BASEDB=<base_db>
- [/WORKSET=<workset_spec>]
- [/[NO]PASSWORD_SAVE]
- [/ATTRIBUTES=(site=<site>,group_id=<group_id>, Dept=<dept>, Full_name=<fullname>, phone_no=<PhoneNo>)]

The CUSR parameters listed below are not used:

- <UserDb>
- /USER | /PROXY [/USER=<user>] | /REPORT /USER
- /DEF_TAB
- /TEMP_TAB
- /[NO]SECURE

XREG – Un-register Users

Dimensions Command

XREG has the following parameters. It is very similar to DUSR, except that there is no user database to remove.

- <UserId>
- /BASEDB=<base_db>
- /[NO]KEEP

MREG – Migrate User

Dimensions Command

MREG has <UserDb> as a parameter. It retains a user's roles and delegations, enables the user to access the base database, and drops the current Oracle user database identity.

Additional Utilities

The database package includes several additional scripts or executables for specialized purposes. These provide the following facilities:

- Readjust Dimensions sequence generator values.
- Create an export file containing the *$GENERIC* process model from a specified Dimensions base database.
- Create a new empty Oracle database.
- Populate a new empty Oracle database with *PCMS_SYS* user data.
- Create a Dimensions database within an existing Oracle database and install generic process model from a specified file.
- Maintain RDBMS statistics for Cost Based Optimization.

Set Dimensions Sequence Values

This executable operates on the sequence generators used in a given base database, and re-initializes each sequence generator to a value in excess of any value used in the base database. It is invoked by the command:

- For UNIX:

  ```
  ${PCMS_DBASE}/set_pcms_seq_value <basedb>/<passwd>
  ```

- For Windows NT/2000:

  ```
  set_pcms_seq_value <basedb>/<passwd>
  ```

  NOTE If the program is not in your path, it can be found in the *DBASE* directory.

This program is called from the *make_basedatabase* utility (see "Create and Initialize a New Dimensions Base Database" on page 113), which is also used during a Dimensions installation. It should also be used after any import of Oracle data into a Dimensions database schema.

Create Generic Export File

This script creates an export file containing the *$GENERIC* process model from a specified base database. This script can be accessed interactively from the Process Modeler (refer to "Export Process Model to a File" on page 87 for further information). It can also be invoked by the command:

- **For Windows NT/2000**

  See "Windows NT/2000 Only" on page 109.

- **For UNIX:**

  ```
  ${PCMS_DBASE}/make_gen_export_file
     [-p <product_id>]
     [-d <default_tablespace>]
     [-t <temp_tablespace>] <basedb> <export_file_spec>
  ```

where

`<basedb>`	is the name of the base database from which a process model is to be exported.
`<export_file_spec>`	is the export file to be generated. This must be in a directory to which the user has write permission.
`<product_id>`	is the name of the product in the base database *<basedb>* whose process model is to be exported. If this is not specified, then the *$GENERIC* product is used by default.
`<default_tablespace>`	is the default tablespace to be assigned to the temporary internal database user *PCMS_TEMP*. If this is not specified, then it assumes the same default tablespace as for *<basedb>*.
`<temp_tablespace>`	is the temporary tablespace to be assigned to the internal temporary user *PCMS_TEMP*. If this is not specified then it assumes the same temporary tablespace as for *<basedb>*.

Notes

1. This script must be run from an account that is enabled to use the Dimensions *CRDB* command, and it is suggested that it be run either on the *pcms* account or by the Tool Manager for the base database whose generic process model is to be exported.
2. The standard Dimensions environment must be set as described General
3. Prerequisites on page 44.

Windows NT/2000 Only

For Windows NT/2000 an alternative method must be used to generate a script to create an export file containing the *$GENERIC* process model from a specified Dimensions base database:

- Create a temporary base database into which the Process Model you wish to export will be copied

```
pcms crdb pcms_tmp /gendb=<basedb>
   /product=<product_id> /user=<user>
```

where

`<basedb>`	is as described for UNIX.
`<product_id>`	is as described for UNIX.
`<user>`	is the name of user running this command. This needs to be *pcms* or another Tool Manager.

- Export the Process Model

```
exp pcms_tmp/pcms_tmp file=<export_file_spec>
parfile=<Dimensions_root\dbase>make_gen_export_file.pars
```

where

`<export_file_spec>` Is as described for UNIX.

- Delete the temporary base database

```
pcms dldb pcms_tmp
```

Create a New Empty Oracle Database

- For UNIX:

  This script creates a new database, containing the standard database users *SYS* and *SYSTEM*, and no others. It is invoked by the command:

```
${PCMS_DBASE}/create_database -o_sid <SID>
   -db_name <db-name> -nls_char <character_set>
   -owner <owner> -group <group> -db_size <data-size>
   -redo_size <redo-size> [-rdb_size <rbs_size>]
   [-temp_size <temp_size>] [-sys_size <system_size>]
   [-db_dir <db_dir>]
```

where

`-o_sid <SID>`	is the *ORACLE_SID* value to be used for the new database
`-db_name <db-name>`	is the database name for the new database
`-nls_char <characterset>`	is the name of the national language characterset in which to create the database in.
`-owner <owner>`	is the login identity of the user who is to own the new database files. This should normally be the same as the owner of the other Oracle files, and is often *oracle*.

`-group <group>`	is the group identity for the new database files. This should normally be the same as for the other Oracle files, and is usually *dba*.
`-db_size <data-size>`	is the size of the *PCMS_DATA* tablespace in the new database, in megabytes. This must be an integer, and its minimum value is 100.
`-redo_size <redo-size>`	is the redo file size for the new database in megabytes. This must be an integer, and its minimum value is 2. Two redo files are created, both of this size.
`-rdb_size <rbs_size>`[a]	is the size of the *PCMS_RBS* tablespace in the new database, in megabytes. This must be an integer, and its minimum value is 30.
`-temp_size <temp_size>`[a]	is the size of the *PCMS_TEMP* tablespace in the new database, in megabytes. This must be an integer, and its minimum value is 10.
`-sys_size <system_size>`[a]	is the size of the *SYSTEM* tablespace in the new database, in megabytes. This must be an integer, and its minimum value is 80.
`-db_dir <db_dir>`[a]	is the directory in which the database files are to be created. If this is not specified, the files will be created in the directory *`${ORACLE_HOME}/dbs`*.

a. Optional parameter.

Notes

1 This script must be run from an account in the same group as the owner of the Oracle installation – normally *dba*.

2 The environment variable *ORACLE_HOME* must be set to an appropriate value. All the database files will be created in the directory *$ORACLE_HOME/dbs*, unless *<db_dir>* is specified. The operation will result in the creation of four tablespaces *SYSTEM*, *PCMS_DATA*, *PCMS_TEMP* and *PCMS_RBS*. See "Use of Tablespaces" on page 137.

- For Windows NT/2000:

  Not currently supported.

Populate a New Database with PCMS_SYS Data

This script transfers standard data into the *PCMS_SYS* account on a new database. This must be done before any Dimensions base databases are created. It is invoked by the command:

- For UNIX:

```
${PCMS_DBASE}/fill_pcms_sys [-u]
   <Oracle-system-password>
   <pcms_sys-password> <PCMS-rel>
```

where

`<-u>`	optional parameter that suppresses the creation of the *PCMS_SYS* user. Only use this parameter if the *PCMS_SYS* user already exists.
`<system-password>`	is the database password for the *ORACLE SYSTEM* user.

`<pcms_sys-password>`	is the database password for the *PCMS_SYS* user who is to be created by this script.
`<PCMS-rel>`	is the Dimensions release number, for example *7.1*.

Notes

1. The environment must include *ORACLE_HOME* and *ORACLE_SID*, the command search path must include the command *hostname*, and the Oracle instance must be running.
2. This script should be run after the *create_database* script, and it will set up the suggested database user in the *SYSTEM* tablespace, and populate its tables with standard Dimensions initial values.
3. *PCMS_SYS* always resides in the *SYSTEM* tablespace, as it is required by all Dimensions users at all times.

- For Windows NT/2000:

  Not currently supported.

Create and Initialize a New Dimensions Base Database

This program creates a database within an existing Oracle database, installs a generic process model from a specified file and optionally enables you to create associated user databases. It is invoked by the command:

```
make_basedatabase <basedb> <system-password>
   <toolman> <export_file_spec>
   [  <DefaultTableSpace> [<TemporaryTableSpace>]
      [<list_of_userdbs_file>]  ]
   [/nopassword_save | /password_save]
```

where

`<basedb>`	is the name of the base database to be created.
`<system-password>`	is the database password for the *SYSTEM* user.
`<toolman>`	is the login identity of the user who is to be Tool Manager for the new base database.
`<export_file_spec>`	is the export file containing the generic process model required for the new Dimensions base database.
`<DefaultTableSpace>`	is the optional parameter for specifying the default tablespace for the database objects. If this *and* *<DefaultTableSpace>* *and* *<list_of_userdbs_file>* are *not* specified, then objects default to the *PCMS_DATA* tablespace.
`<TemporaryTableSpace>`	is the optional parameter for specifying the temporary tablespace for temporary segments. If this *and* *<TemporaryTableSpace>* *and* *<list_of_userdbs_file>* are not specified, then temporary segments default to *PCMS_TEMP*. However, to make use of the *<TemporaryTableSpace>* positional parameter you *must* first specify an entry for *<DefaultTableSpace>* (*PCMS_DATA* is the normal value).

`<list_of_userdbs_file>` is the optional parameter for specifying a text file listing the user databases to be created in association with the newly created base database. Entries must be formatted as two-columned rows, with each column separated by white space. The first column for each entry specifies the operating system username[a] and the second column specifies the related Oracle username, for example:

```
user2   basedb2_user2
user3   basedb2_user3
```

To make use of the *<list_of_userdbs_file>* parameter you *must* first specify an entry for *<DefaultTableSpace>* (*PCMS_DATA* is the normal value) and *<TemporaryTableSpace>* (*PCMS_TEMP* is the normal value).

a. *make_basedatabase* does *not* attempt to create the operating system usernames. Also, it does not necessarily need them to pre-exist in order to execute (you can create them afterwards).

Certain Dimensions PC client tools[4] have a "remote automatic login" facility (see, for example, the *PC Client User's Guide*). To be able to utilize this facility, the user's operating system login password on the remote Dimensions server (UNIX or Windows NT/2000) must be saved locally on the client PC. To enable the remote automatic login for the Tool Manager, the */PASSWORD_SAVE* qualifier must be specified. By default, this facility is not enabled and can also be explicitly disabled by specifying the */NOPASSWORD_SAVE* qualifier. Please refer to

4. Dimensions Make, PC Client, PVCS Project Merge and SCC Interface

"CUSR – Create Dimensions User Database" on page 92 for further details of client automatic remote logins.

Notes

1 This program may be run from any suitable account, but it is recommended that it should be run from the account that is to be the Tool Manager for the new Dimensions base database.

2 To run this program the standard Dimensions environment must be set as described "General Prerequisites" on page 44.

3 The program should only be run against an Oracle database that has either had the *fill_pcms_sys* script run against it, or has had Dimensions installed into it.

Recreate all Indexes Script

This script examines all the indexes associated with a base database, and generates a further script that will recreate and resize these indexes.

It is invoked by the command:

- For UNIX:

```
sqlplus <basedb>/<passwd>
  @${PCMS_DBASE}create_pcms_index_script
```

- For Windows NT/2000:

```
sqlplus <basedb>/<passwd>
  @<pcms_dbase>\create_pcms_index_script
```

(where

`<pcms_dbase>` is path of the *dbase* directory in the Dimensions"root" directory)

This generates an SQL script file called *create_pcms_indices.sql* that contains the necessary SQL commands to recreate all the indexes on the base database, together with temporary disablement of certain constraints. It also generates two temporary files, called *listtabs.sql* and *recreateindxs.log*. All these files are generated in the current working directory, to which the user must have write access.

The resulting script, *create_pcms_indices.sql* may then be run by the command (for UNIX and Windows NT/2000):

```
sqlplus <basedb>/<passwd> @create_pcms_indices
```

Notes

1. *create_pcms_index_script.sql* – the first part of this procedure – may take some time to run, but there is nothing to prevent this being done while Dimensions is operating normally.
2. *create_pcms_indices.sql* – the second part of this procedure – will *however take* a considerable time to run, depending on the size of the base database. Furthermore, this should not be run during normal operation of Dimensions. The procedure involves the dropping and recreation of indexes and the performance of Dimensions will be affected by the absence (even though temporary) of any of the indexes.

Recreate Selected Indexes Script

If you wish to re-create only specific indexes, you can use the script `recreateindxs_sql` as detailed below.

This script takes an index name as a parameter and directs the SQL statement needed to re-create this index to standard

output. You can then use this SQL statement to regenerate the index that you specified.

- For UNIX:

```
sqlplus <basedb>/<passwd>
   $PCMS_DBASE/dbase/recreateindxs.sql
   <index_name> > <output_file>
```

- For Windows NT/2000:

```
sqlplus <basedb>/<passwd>
   %PCMS_ROOT%\dbase\recreateindxs.sql
   <index_name> > <output_file>
```

where

`index_name`	is the identity of the index to be re-created
`output_file`	is the output file identity

Maintaining Oracle Database Statistics for Performance Optimization

By default, Dimensions 7.x uses Cost-Based Optimization (CBO) to process SQL statements. CBO is an optimization strategy that aims for the best *throughput* of SQL statements when considering how to access data. A vital part of this decision making process involves meta-data called *statistics*. This meta-data provides the Optimizer with information regarding the cost in time and resources of executing various different data access plans, and from this information the most efficient plan is calculated. However, for the Optimizer to be able to make reliable decisions this statistical meta-data must be kept up to date. To aid the DBA with this task, Dimensions 7.x provides a new utility that helps maintain these statistics.

This utility is invoked as follows:

```
pcms_dbstats [-e\c\s\d] <database-name>
```

where

`-e`	will *estimate* the statistics for a specified database. This is the default option.
`-c`	will *calculate* the statistics for a specified database.
`-s`	will *sample* the statistics for a specified database.
`-d`	will *delete* the statistics for a specified database. This will remove all statistics meta-data for that database.
`<database-name>`	is the database name for which statistics are to be generated. This database must be specified in the format <*dbname/dbpasswd*>.

Notes

1 The utility supports invocation on either single databases or the entire Oracle instance. If you specify *SYSTEM* as the database upon which to run the utility, then *all* Dimensions base databases will have *estimate* statistics generated. This type of invocation on *SYSTEM* only supports generating *estimate* statistics (other options will be ignored).

2 This utility does not support encrypted passwords. You must specify full <*dbname/dbpasswd*> connection details.

Types of Statistic

The utility supports various different mechanisms for gathering statistics. These are:

- **Estimate Statistics**

  These are created based on a random sample of data taken from a specific database object. From these estimates the optimizer extrapolates the data spread in that object.

- **Sample Statistics**

  These are similar to estimated statistics, except that 20% of the contents of the database object are taken into consideration when extrapolating the data spread in that object.

- **Computed Statistics**

  These take longer to generate, but are exact statistics about what is contained in that object.

Notes

1. By default the utility will generate estimate statistics. These will usually be sufficient for the optimizer to select an optional data access path.
2. When a Dimensions base database is created, estimate statistics are automatically generated.

Recommendations on when to Run the Utility

For optimal database performance, it is recommended that this utility be run on a regular basis. Just how often to run it will depend the actual amount of database activity that is undertaken, but typically twice a month should be sufficient.

6 Recommendations

In this Chapter

Backup and Recovery

NOTE You are strongly advised to back up your Dimensions Oracle database and all Dimensions item libraries on a regular basis (preferably daily). Furthermore, you must ensure that the database and the item libraries are backed up as a consistent set, representing a valid snapshot of your Dimensions data at a specific point in time. In the unlikely event of a database failure, or disk failure, where a full recovery to a consistent Dimensions state is not possible, MERANT will assist you to resolve database problems once you have restored all relevant data files from your latest backup. In accordance with our standard maintenance terms, we will reserve the right to charge for such assistance and will do so if a comprehensive backup is not available, or is inconsistent, or is considered out of date.

This chapter outlines a number of backup recommendations that you should follow to ensure that your database and your Dimensions installation can be fully and consistently be recovered.

The information stored under Dimensions which should regularly be backed up as a protection against loss of data comprises:

- Information stored within base databases.
- Project files (i.e. product-items) stored within the Dimensions item libraries. The item libraries are operating-system directories.

NOTE To determine the nodes and directory names for each of the item libraries connect to each base database in turn (base databases are listed with Dimensions *LSDB*, see page 75) and execute the following SQL*Plus commands (see "Access to the DBA Utilities" on page 46 regarding command line interfaces).

```
% SQL*Plus pcms/pcms
SQL> spool listing.out
SQL> select distinct node_name, library from obj_types;
SQL> exit
```

(Replace *listing.out* with your own filename in which to store the information displayed to the screen.)

Several methods of backup and restore are available. The method and frequency of backups to be used are left to the discretion of the system manager and DBA for each installation; however, MERANT recommends daily backups are taken.

In general, users are recommended to follow one of three strategies:

1. If it is not acceptable to lose any data in the event of a problem, then the Dimensions database should be run in *ARCHIVELOG* mode.
2. If it is acceptable to lose a limited amount of data in the event of a disk failure, then the Dimensions database can be operated in *NOARCHIVELOG* mode. This avoids the overhead of archiving filled online redo log files.
3. If the database is required 24 hours, seven days a week and 365 days a year, then the Dimensions database should *not* be operated in *NOARCHIVELOG* mode as this would require full database backups to be taken while the database is shut down. To meet this round-the-clock requirement there should also be consideration made of disk shadowing for the Oracle database files. Disk shadowing should *not* have a "write forward cache enabled" as this may give recovery problems. With disk shadowing the requirement for *ARCHIVELOG* mode is reduced only slightly, as the shadow disk could mirror the same corruption as the primary disk.

For detailed information on how to best implement *ARCHIVELOG* mode for your Oracle installation please contact the PVCS Support Center.

Performing a Full Backup

A full backup is suitable for users who can temporarily shut down their the Dimensions database. It can be taken if the database is operating in either *ARCHIVELOG* or *NOARCHIVELOG* mode.

- Note down the locations of all database files.

```
$ svrmgrl
SVRMGR> spool listing.out
SVRMGR> connect internal/<sys_password>[1];
SVRMGR> SELECT name FROM sys.v$datafile;
SVRMGR> SELECT member FROM sys.v$logfile;
SVRMGR> SELECT * FROM sys.v$controlfile;
SVRMGR> SHOW PARAMETER control_files;
SVRMGR> exit
```

- Shut down the database with normal priority (see "Starting and Stopping Oracle" on page 133).
- Perform the backup using operating-system utilities and restart the database (see "Starting and Stopping Oracle" on page 133). Although this approach is considerably quicker than using the Oracle *EXP* and *IMP* utilities (see "Full System Exports" on page 128), integrity is *not* guaranteed. Both methods should be employed.

1. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

Performing Partial Backups

For Online Tablespace and Online Data File Backups

This is suitable for users who use the *SYSTEM* tablespace or any named tablespace that, for one reason or another, cannot be taken offline.

- Identify the data files.

```
$ svrmgrl
SVRMGR> spool listing.out
SVRMGR> connect internal/<sys_password>[2]
SVRMGR> select tablespace_name, file_name
     2> from sys.dba_data_files
     3> where tablespace_name = '<Tablespace>';
SVRMGR> spool off
```

NOTE *<Tablespace>* is any named tablespace, including *SYSTEM*, and is case-sensitive.

- Mark the beginning of the online tablespace backup.

```
SVRMGR> alter tablespace <Tablespace> begin backup;
```

NOTE While the backup is in progress you cannot take the tablespace offline normally, shutdown the instance, or begin another backup of the tablespace.

- Backup – using operating-system utilities – the online data files identified in the earlier step.

2. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

- During the operating-system backup, the status of the online data files can be viewed.

```
SVRMGR> select v.file#, v.status, d.name
     2> from sys.v$backup v, sys.v$datafile d
     3> where v.file# = d.file#;
```

A *NOT ACTIVE* status indicates that the file is not being backed up.

An *ACTIVE* status indicates that the file is currently being backed up.

- Once the operating-system backup has completed, return to the "SVRMGR Command" session and mark the end of the online tablespace backup.

```
SVRMGR> alter tablespace <Tablespace> end backup;
SVRMGR> exit
```

For Offline Tablespace and Offline Data File Backups

This is suitable for, and preferred by, users who are able to take tablespace(s) offline. As the *SYSTEM* tablespace *cannot* be taken offline, the online procedure just described must be used to backup that tablespace.

- Identify the data files of the offline tablespace(s).

```
$ svrmgrl
SVRMGR> spool listing.out
SVRMGR> connect internal/<sys_password>[3]
SVRMGR> select tablespace_name, file_name
     2> from sys.dba_data_files
     3> where tablespace_name = '<Tablespace>';
SVRMGR> spool off
```

3. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

NOTE *<Tablespace>* is any named tablespace and is case-sensitive.

- Take the tablespace(s) offline, using *normal* priority if possible.

```
SVRMGR> alter tablespace <Tablespace> offline;
```

- Backup–using operating-system utilities–the offline data files identified in the earlier step.
- Once the operating-system backup has completed, return to the "SVRMGR Command" session and bring the tablespace(s) back online.

```
SVRMGR> alter tablespace <Tablespace> online;
SVRMGR> exit
```

NOTE Typically product related data is located in the *PCMS_DATA* tablespace. However the *PCMS_SYS* data is *still* in the *SYSTEM* tablespace.

Recovering from Failure during Backup

Users will not need to recover previous backups, but may require archived redo logs. Querying the table *V$BACKUP* will display if files were in the process of being backed up (see page 126).

Recovery is performed with following command:

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[4]
SVRMGR> startup open recover
SVRMGR> exit
```

4. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

Control File Backups

After any structural changes to the database, adding of new data files etc., users must backup the Control File if operating in *ARCHIVELOG* mode.

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[5]
SVRMGR> alter database backup controlfile
     2> to 'filename' reuse;
```

where'l

`<filename>`	is a fully specified filename that indicates the filename for the new control file backup.

Full System Exports

In addition to operating-system image backups, it is strongly advisable to take *SYSTEM* exports of the RDBMS database. A RDBMS *SYSTEM* export copies all the information in the database (including the Dimensions base databases) to a file. This ensures that every byte of information is accessible and therefore informs the DBA of any problems that may otherwise have been hidden. Another advantage of taking RDBMS *SYSTEM* exports is that the export files can be used for importing back the exported data into the database, hence making it possible to go back to the database state at the time of the export. Exporting and importing can often be combined with regular reorganization of the database to recover unused space (see "SPAC – Display RDBMS Database Space Usage" on page 57).

5. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

If you wish to perform an RDBMS *SYSTEM* export using a command file, then you can use the Oracle Export utility, see "EXP – Oracle Export Utility" on page 78). An example is as follows:

- For UNIX (Bourne Shell):

```
% $ORACLE_HOME/bin/exp userid=system/<password> \
full=y grants=y file=export_tue_1830.exp compress=y 1> \
export_tue_1830.log 2>&1
```

- For Windows NT/2000:

```
X:> exp userid=system/<password> full=y grants=y
   compress=y file=export_tue_1830.exp
   log=export_tue_1830.log
```

In these examples: *export_tue_1830.exp* is the full system export file created; and *export_tue_1830.log* is the log file into which is redirected any messages that would have been outputted to the terminal window as a consequence of the export operation. It is *essential* that you check the log file for any errors that may have occurred.

You can also use the Oracle Import utility, (see "IMP – Oracle Import Utility" on page 80), to import back exported data into the database. For example, the import operation corresponding to the above export is:

- For UNIX:

```
% $ORACLE_HOME/bin/imp userid=system/<password> \
full=y grants=y file=export_tue_1830.exp commit=y \
recalculate_statistics=y
```

- For Windows NT/2000:

```
X:> imp userid=system/<password> full=y grants=y
   file=export_tue_1830.exp commit=y
   recalculate_statistics=y
```

IMPORTANT POINTS

1 You should import an RDBMS *SYSTEM* export only if the database is re-created and *empty*. Failure to do so may result in duplication of your data.

2 After a database export, you should always check the log file for any errors that may have occurred.

3 For a RDBMS *SYSTEM* export of the database, it is strongly recommended that the database is shutdown and restarted in *restricted* mode and that no other access is attempted until the export is complete. This can be achieved from a "SVRMGR Command".

4 If an operating-system backup is used for the RDBMS database files, it is absolutely vital that the database is shut *completely* before the backup begins, and that no start up is attempted before the backup completes. If this is not observed, it may prove impossible to use the backup files for recovery purposes.

5 The use of the *NLS_LANG* environment variable will affect the national character set in which the database export is performed. If you set this variable to a different value than that in which the RDBMS database was created, then your exported data *will be* subjected to character set conversion. If this is not desirable then ensure that this environment variable is *not* defined.

6 To preserve consistency between the database and the item libraries after recovery, it is necessary to take backups of both of these simultaneously: i.e. export or backup all your database files and the associated product item libraries at the same time. This ensures physical synchronization among all the components.

Additional Control Files

Every Oracle 8i instance uses a small binary file called a *Control File*. This file contains the information required to startup the Oracle Instance. This file is resident on:

- UNIX

  In the directory *$ORACLE_HOME/dbs* – it is given the name *cntrl<db_name>.dbf.*

- Windows NT/2000

  In the *orant\database* directory – it is given the name *ctl1<db_name>.ora*.

It is recommended that multiple copies of this file are kept on separate disks to reduce the risk of a media failure removing the control files. The UNIX and Windows NT/2000 installations, however, create only one. To create another Control File (second one for UNIX or Windows NT/2000) and/or to relocate the existing or created additional copies of the Control File (UNIX or Windows NT/2000) use the following procedure:

1 Shut the database down.

  See "Starting and Stopping Oracle" on page 133.

2 Copy the Control File to another disk.

  - For UNIX

    Modify the file *init<db_name>.ora* by adding or modifying the parameter *CONTROL_FILES* to include the name of the new Control File. The file *init<db_name>.ora* is located in the *$ORACLE_HOME/dbs* directory.

For example, suppose that the Oracle SID is *ora8i*, the existing Control File is:

```
/disk1/oracle/dbs/cntrlora8i.dbf
```

and this file is copied into a new file called:

```
/disk2/oracle/data/cntrl2ora8i.dbf
```

residing on a different disk.

To take into account the new Control File, add the following line to the file *initora8i.ora:*

```
CONTROL_FILES =
(/disk1/oracle/dbs/cntrlora8i.dbf,
/disk2/oracle/data/cntrl2ora8i.dbf)
```

- For Windows NT/2000:

  Modify the file *init<db_name>.ora* by adding or modifying the parameter *CONTROL_FILES* to include the name of the new Control File. The file *init<db_name>.ora* is located in the *orant\database* directory.

  The *editing* process in analogous to that described for UNIX.

3 Restart the database.

See "Starting and Stopping Oracle" on page 133.

Starting and Stopping Oracle

General

Many DBA operations can be performed while Oracle and Dimensions are running normally. However there are some functions during which other users should be prevented from accessing the database. This implies that Oracle must be shut down and restarted in a *restricted* mode. This section describes how to do this.

The correct environment must be set up as described in "General Prerequisites" on page 44.

All startup and shutdown operations should be done from an account that meets the prerequisites of the appropriate operating system (as described in the subsections of "General Prerequisites" on page 44).

Use of the Dimensions Scripts

If Oracle is running and Dimensions is in use, the best way of shutting down Oracle is to use the script provided in the *pcms_mmi* subdirectory.

First ensure that nobody is using Dimensions (or Oracle), and then enter the command:

- **For UNIX, as root:**

```
sh ${PCMS_MMI}pcms_shutdown
```

An optional parameter may be supplied to the above as follows:

- NORMAL — Dimensions will wait for all connected users to disconnect from the Oracle database. This is the default behavior when no parameter is specified.
- IMMEDIATE — Dimensions will not wait for connected users, but will rollback current transactions. Any recovery on subsequent startup is automatic. *IMMEDIATE* is recommended if the machine is going to be shut down, or if the database is malfunctioning.
- ABORT — Dimensions will shut down the Oracle database instantaneously. *ABORT* should *only* be used if neither of the above methods succeed in shutting down the database.

- **For Windows NT/2000:**

```
net stop PVCS_Dimensions_Service [6]
net stop PVCS_Dimensions_Listener_Service
net stop PVCS_Replicator [6]
net stop PVCS_Dimensions_Licensing [7]
net stop OracleDimensionsTNSListener
net stop OracleService<sid>
net stop OracleDimensionsClientCache [6]
net stop OracleDimensionsAgent [6]
net stop OracleDimensionsDataGatherer [6]
```

At the conclusion of DBA operations, Oracle and Dimensions should be restarted by the script provided in the *pcms_mmi* subdirectory. Enter the command:

---

6. Stop this service if present and running.
7. This is the FLEX*lm* service used by Dimensions and may be named differently if configured manually.

- **For UNIX, as root:**

```
sh ${PCMS_MMI}pcms_startup
```

- **For Windows NT/2000:**

```
net start OracleDimensionsTNSListener
net start OracleService<sid>
net start PVCS_Dimensions_Service [8]
net start PVCS_Dimensions_Listener_Service
net start PVCS_Replicator [8]
net start PVCS_Dimensions_Licensing [9]
```

Stopping Oracle During DBA Operations

To stop Oracle during DBA operations, or if the Dimensions shutdown script fails (UNIX), use SVRMGRL commands as follows:

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[10]
SVRMGR> shutdown <option>
SVRMGR> exit
```

This should be done first of all with *<option>* omitted. In case of failure, first check that the *TWO_TASK* environment variable (UNIX) or *LOCAL* environment variable (Windows NT/2000) is not inadvertently set to a different Oracle instance. If users are still connected, this command will hang until those users have disconnected. If it is not possible to wait, then try again with <option> set to *IMMEDIATE*, or after repeated failures with <option> set to *ABORT*.

---

8. This service may not exist on all systems.
9. This is the FLEX*lm* service used by Dimensions and may be named differently if configured manually
10. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

Starting Oracle in Restricted DBA Mode

To start up Oracle in *restricted* DBA mode, for use during some the Dimensions DBA operations, use *SVRMGRL* commands as follows:

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[11]
SVRMGR> startup restrict open <db_name> exclusive
SVRMGR> exit
```

This starts Oracle but only allows users with *RESTRICTED SESSION* privilege to connect to the database. To start Oracle for general use, proceed as above, but omit the *RESTRICT* keyword.

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[11]
SVRMGR> startup open <db_name> exclusive
SVRMGR> exit
```

In the event of the database refusing to start it is possible to issue a stronger command:

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[11]
SVRMGR> startup force open <db_name> exclusive
SVRMGR> exit
```

It is then a good idea to shut the database down and restart it normally.

---

11. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

Use of Database Roles

Access to base database tables is controlled via the use of database roles. Two roles (*<basedb>_USER* and *<basedb>_REPT*) are created for each base database, and access rights for base database tables are granted to these roles. A user created by *CUSR* is then granted one of these roles, rather than being granted privileges on all the base tables and views themselves.

The *<basedb>_USER* role is granted limited privileges on base tables, comprising delete, insert, select and update privileges, but excluding alter, index and references privileges. Hence users with this role may examine and modify table data, but they cannot modify any table definitions. This role is also granted select privilege on all the Dimensions views (both in-built views and those provided for reporting such as the Dimensions Published Views). This role is given to all normal and */PR0XY* users.

The *<basedb>_REPT* role is granted only select privilege on a limited set of views - including those provided for reporting such as the Dimensions Published Views - but excluding in-built views. This role is given to report users, and serves to restrict their access to the Dimensions tables. In particular report users cannot see the contents or structure of base tables except through the views. As they only get select privilege they cannot insert or modify table data.

Use of Tablespaces

The Dimensions database administration functions and installation script allow the use of database tablespaces other

than the *SYSTEM* tablespace. This has many advantages, for example:

- Separation of database control information from the Dimensions data.
- More effective backup strategies.
- Allocation of database files on different devices to improve performance.
- Minimization of database fragmentation.

When Dimensions is installed, tablespaces for the following are created in the UNIX *$ORACLE_HOME/dbs* or Windows NT/2000 *orant\database* directory (or other directory specified during installation) with the approximate sizes in Mbytes indicated.

SYSTEM	150	PCMS_TEMP	50
PCMS_RBS	50	PCMS_DATA	500[a]

a. Refer to "Importing the Entire System Database" on page 82 for further details.

Refer to the *README* file for the latest space requirements.

The *SYSTEM* tablespace is used to hold the data dictionary and the tables for the *PCMS_SYS* database user. The *PCMS_RBS* tablespace is used for rollback information for all database users.

Initially, all the Dimensions users are set up with their database tables in the *PCMS_DATA* tablespace, and with *PCMS_TEMP* as their temporary tablespace. A major advantage of using a separate temporary tablespace is that this prevents large queries and similar operations from fragmenting the space used for allocation to tables.

We recommend that all the Dimensions users (except for the special *PCMS_SYS* user) are set up to use non-*SYSTEM* tablespaces both as their default tablespace and as their temporary tablespace. There are many advantages to be had from using

separate tablespaces for different base databases (particularly when these are large), but it is preferable for all the user databases to use the same tablespaces as the corresponding the base databases (this is the default behavior of the *CUSR* function).

Further information on how to create and move tablespaces is given in "Tablespaces" on page 161. To increase the size of a tablespace, use the *INCR* command which is described in "INCR – Increase the RDBMS Database Space" on page 62.

7 Dimensions Quick Tuning Guide

In this Chapter

Introduction

Oracle RDBMS Version 8i embeds features that can be tuned by the DBA to improve the performance of an Oracle Instance database system. These features include:

- Shared pool area
- Dictionary cache
- Buffer cache
- Rollback segments
- Redo buffers.

We recommend the DBA to take advantage of these features so as to achieve better Dimensions performance from the database. This chapter outlines some practical steps to this end.

Diagnostic Scripts to Analyze your Database

Several diagnostic scripts are provided to collect statistics about your installation. To fully interpret the results and before taking action you may need to consult your Dimensions support facility – the PVCS Support Center. The SQL*Plus scripts can be entered via various Oracle Command Line Interfaces – see "Access to the DBA Utilities" on page 46.

pcms_snap.sql

The diagnostic script *pcms_snap.sql* is used to collect statistics relevant to the tuning of the RDBMS.

- For UNIX:

```
sqlplus system/<system-password> @${PCMS_DBASE}pcms_snap
```

- For Windows NT/2000:

```
sqlplus system/<system-password> @<pcms_dbase>\pcms_snap
```

Where <*system-password*> is the Oracle password for the *SYSTEM* user and <*pcms_dbase*> is the path of the *dbase* directory in the Dimensions "root" directory.

This script places its output in a file *pcms_snap.list* in the current working directory (to which you must have write access).

Notes

1 The *pcms_snap.sql* script uses the following Oracle system views that may not be defined initially in a Dimensions database:

   - DBA_LOCKS
   - DBA_DML_LOCKS
   - DBA_DDL_LOCKS.

2 These views should be created before you run the *pcms_snap.sql* script for the first time, thus:

   - For UNIX

```
sqlplus sys/<password>
start $ORACLE_HOME/rdbms/admin/catblock
exit
```

- For Windows NT/2000

  (First check the location of the *catblock.sql* file, and modify the start command correspondingly.)

```
sqlplus sys/<password>
start
c:\Win32App\PVCS\Dimensions\orant\rdbms\admin\catblock
exit
```

3 This should take only a few moments, and the existence of these views will not affect database performance, nor do they require any significant amount of storage. Note that the *catblock.sql* script is to be run by the *SYS* database user, not by *SYSTEM*.

oracle_sizing.sql

The diagnostic script *oracle_sizing.sql* gives information about the Oracle database as a whole.

- For UNIX:

```
sqlplus system/<system-password>
@${PCMS_DBASE}oracle_sizing
```

- For Windows NT/2000:

```
sqlplus system/<system-password>
@<pcms_dbase>\oracle_sizing
```

Where <*pcms_dbase*> is the path of the *dbase* directory in the Dimensions "root" directory)

This script places its output in a file *oracle_db_sizing.lst* in the current working directory (to which you must have write access).

diskact.sql

The diagnostic script *diskact.sql* list all the relevant disk files that are used by your Oracle instance. Your can run this script as follows:.

- For UNIX:

```
sqlplus system/<system-password> @${PCMS_DBASE}diskact
```

- For Windows NT/2000:

```
sqlplus system/<system-password>
@<pcms_dbase>\diskact
```

Where <*pcms_dbase*> is the path of the *dbase* directory in the Dimensions "root" directory)

This script places its output in a file *oracle_db_sizing.lst* in the current working directory (to which you must have write access).

pcms_sizing.sql

The diagnostic script *pcms_sizing.sql* gives information about individual Dimensions base databases. You can connect to *sqlplus* as any user, but you need the base database password to run the command. If necessary you may need to use Dimensions *CPAS* (on page 98) to give the base database a non-secure password while you run the command.

- For UNIX:

```
sqlplus system/<system-password>
SQL> start ${PCMS_DBASE}pcms_sizing <basedb>
   <basedb-password>
```

- For Windows NT/2000:

```
sqlplus system/<system-password>
SQL> start <pcms_dbase>\pcms_sizing <basedb>
  <basedb-password>
```

Where *<pcms_dbase>* is path of the *dbase* directory in the Dimensions "root" directory)

This script places its output in a file *sizing_<basedb>.lst*, and it also generates a temporary file *tabs_<basedb>.sql*. Both of these are in the current working directory, to which you must have write access.

Oracle Version 8i Runtime

Oracle Version 8i database files, as set up by Dimensions installation, typically include seven major files: the data file for the *SYSTEM* tablespace, data files for the *PCMS_DATA*, *PCMS_TEMP* and *PCMS_RBS* tablespaces, two redo log files and a Control File. At installation (unless otherwise specified), they are all placed in the *$ORACLE_HOME/dbs* directory (UNIX) or in the *orant\database* directory (Windows NT/2000).

The *SYSTEM* tablespace data file contains the data dictionary for the database, and also contains the data for the *PCMS_SYS* user (see "PCMS_SYS Account" on page 41).

The *PCMS_DATA* data file contains the actual Dimensions data. The *PCMS_TEMP* data file contains temporary segments to act as workspace for Dimensions users, while the *PCMS_RBS* data file contains the rollback segments.

The redo log files protect the database in the event of a system crash by recording changes that were not written to the database. Changes are written sequentially by the log writer *LGWR* process.

The Control File is a small binary file containing information about the database that is required at start up. As discussed in "Additional Control Files" on page 131, multiple copies of these Control Files are recommended to be kept on separate disks to reduce the risk of a media failure removing the Control Files.

For every Oracle Instance, there exists a text file called *init<db_name>.ora* for UNIX and for Windows NT/2000.

This text file contains RDBMS configuration parameters that are used each time the instance is started up. These parameters include operating-system dependent parameters such as the number of data buffers in main memory (*DB_BLOCK_BUFFERS*) and the size of each block (*DB_BLOCK_SIZE*), etc. As discussed below, some of these parameters can be chosen carefully so as to improve your system's performance.

Dimensions provides a default *init.ora* file at installation. The content of this file is presented in the relevant Dimensions Installation Guide, and some aspects are discussed in "init.ora Parameters" on page 148.

Several background processes (Windows NT/2000 "services") run when an Oracle Instance, say *ora8i*, is active. At a minimum they comprise:

- For UNIX:

```
ora_pmon_ora8i      (Process Monitor Process)
ora_dbw0_ora8i      (Database Writer Process)
ora_lgwr_ora8i      (Log Writer Process)
ora_smon_ora8i      (System Monitor Process)
ora_ckpt_ora8i      (Checkpoint Process)
```

- For Windows NT/2000:

```
OracleServiceORA8i
OracleDimensionsTNSListener[1]
```

1. When connecting from another machine.

These processes/services bind themselves to a shared memory region called the *SGA* (System Global Area). Its size is governed mainly by the *init.ora* parameters *DB_BLOCK_BUFFERS* and *DB_BLOCK_SIZE*.

An Oracle RDBMS database contains one or more Rollback Segments. These segments form part of the database operating-system files and are generally used to provide transaction rollback and recovery. When an RDBMS database is created for the first time, an initial rollback segment, called *SYSTEM*, is created in the *SYSTEM* tablespace. The installation of Dimensions will create four additional rollback segments in the *PCMS_RBS* tablespace – each of initial size 100 Kbytes (UNIX) and 1 Mbyte (Windows NT/2000) – plus one small rollback segment in the *SYSTEM* tablespace (which is normally offline, but is provided for database maintenance purposes).

init.ora Parameters

The setting of the *init.ora* parameters customizes the performance of the Oracle Instance. While the defaults provided by the Dimensions Installation often yield an adequate default performance, as your use of Dimensions grows it will be necessary to maintain performance by careful tuning each instance parameter.

How to Implement Changes to init.ora Parameters

The manner of implementing changes to the *init.ora* parameters varies depending on whether the operating system is UNIX or Windows NT/2000.

NOTE For UNIX and Windows NT/2000, the changes to *init.ora* parameters only take effect after the Oracle Instance has been stopped and restarted.

Implementing Changes to init.ora Parameters for UNIX Systems

Changes to *init.ora* parameters can be made by editing the *$ORACLE_HOME/dbs/init<db_name>.ora* file appropriate to the Oracle Instance concerned.

Implementing Changes to init.ora Parameters for Windows NT/2000 Systems

Changes to *init.ora* parameters can be made by editing the *orant\database\init<db_name>.ora* file appropriate to the Oracle Instance concerned. The database directory is located in the Dimensions root directory.

Tuning the Data Buffer Cache

DB_BLOCK_BUFFERS (one buffer contains one Oracle block) stores the most recently used data from indexes, tables, views etc. Performance is enhanced dramatically when requested data is found in the buffer cache. The diagnostic script *buffercache.sql* assists in determining a suitable value for

this parameter. However, when increasing this parameter be aware that it may cause paging.

The SQL*Plus scripts can be entered via various Oracle Command Line Interfaces – see "Access to the DBA Utilities" on page 46.

- For UNIX:

```
sqlplus system/<system-password>
@${PCMS_DBASE}buffercache
```

- For Windows NT/2000:

```
sqlplus system/<system-password>
@<pcms_dbase>\buffercache
```

Where *<system-password>* is the Oracle password for the *SYSTEM* user and *<pcms_dbase>* is the path of the *dbase* directory in the Dimensions "root" directory).

This script will return three values based on the buffer cache statistics. You can use these values to determine the "Buffer Cache Hit Ratio" as follows:

```
BufferCacheHitRatio = 1 –(physical reads |
   <db block gets + consistent gets>)
```

Tuning this parameter may mean an SGA larger than the default. Contact the PVCS Support Center should you desire to do this. It should not be necessary to increase *DB_BLOCK_BUFFERS* except in exceptional cases.

Tuning Shared Memory Pool

The Shared Pool is used to contain information relating to previously parsed and executed SQL statements. Performance of SQL parsing operations and calculating data access can be significantly affected by tuning the parameter

SHARED_POOL_SIZE. To determine if this parameter needs adjustment, run the *pcms_snap.sql* script as previously described and contact the PVCS Support Center.

Tuning Redo Log Files

Increase the value of *LOG_CHECKPOINT_INTERVAL* to reduce the frequency of checkpoints. This is the interval at which the redo log file is written to disk (those blocks that have been modified since the last one). Increasing the size of the redo log files to 5 or 10 Megabytes may also help. The diagnostic script *pcms_snap.sql* assists in determining a suitable value for this parameter.

In general, this hit ratio should be greater than 90%. If this value is low, about 60%-70%, then you should consider increasing the value of *DB_BLOCK_BUFFERS*.

The SQL*Plus scripts can be entered via various Oracle Command Line Interfaces – see "Access to the DBA Utilities" on page 46.

- For UNIX:

```
sqlplus system/<system-password> @${PCMS_DBASE}pcms_snap
```

- For Windows NT/2000:

```
sqlplus system/<system-password> @<pcmsdbase>\pcms_snap
```

Where <*system-password*> is the Oracle password for the *SYSTEM* user and <*pcmsdbase*> is the path of the *dbase* directory in the Dimensions "root" directory)

This script places its output in a file *pcms_snap.list* in the current working directory (to which you must have write access).

Tuning Rollback Segments

The SQL*Plus scripts can be entered via various Oracle Command Line Interfaces – see "Access to the DBA Utilities" on page 46.

- For UNIX:

  The parameter *ROLLBACK_SEGMENTS* defines a list of private rollback segments used when running the database instance. Dimensions installation sets this parameter thus:

  ```
  rollback_segments = (r01, r02, r03, r04)
  ```

  This parameter must be modified if further rollback segments are added (see "The SQL*Plus scripts can be entered via various Oracle Command Line Interfaces – see "Access to the DBA Utilities" on page 46." on page 152).

  The small rollback segment 'r0' defined in the *SYSTEM* tablespace for database maintenance purposes must not be added to this parameter.

- For Windows NT/2000:

  The parameter *ROLLBACK_SEGMENTS* defines a list of private rollback segments used when running the database instance. Dimensions installation sets this parameter thus:

  ```
  rollback_segments = (r01, r02, r03, r04)
  ```

  This parameter must be modified if further rollback segments are added (see "The SQL*Plus scripts can be entered via various Oracle Command Line Interfaces – see "Access to the DBA Utilities" on page 46." on page 152).

  The small rollback segment 'r0' defined in the *SYSTEM* tablespace for database maintenance purposes *must not* be added to this parameter.

Redo Log Files

A Dimensions Installation will place all Oracle database files on the one disk. Performance, however, can be improved by placing the redo log files on separate disks from the operating-system datafiles.

By default, there are two redo log files, written to sequentially by the *LGWR* process. The *SVRMGR* Command Line Interface provides the necessary commands or dialogs to move these files.

Redo log files are used by Oracle in a cycle, and at any one time when Oracle is running one of the log files is in use. When Oracle is shutdown and restarted the next log file is used in the cycle. When there are two log files the use therefore alternates.

To move the redo log files the *SVRMGR* Command Line Interface is used first to add a new log file into the cycle and then secondly to drop an old log file so that it is not in the cycle. Note that *add* actually physically creates the log file, but *drop* does *not* delete a log file.

It is possible that when attempting to drop that a log file a message will be received:

```
ORA_01515: error dropping log group; no such log
```

By shutting down and starting up Oracle, the next log file in the cycle will be used and the drop should then be successful.

The procedures for moving a redo log file is different for UNIX and Windows NT/2000, and are documented separately in the next two subsections. Note that for UNIX and Windows NT/2000 the file names are enclosed in single quotation marks which *must* be present when using command mode. The Oracle is shown being started up in a special way without the database being opened. After this special way of starting there will be a message received on shutting (see below) which can be ignored:

```
ORA-01109: database not open
```

UNIX Move Log File

The following is an example showing a logical move of a log file named

```
'/usr/oracle/dbs/oldlog.dbf'
```

to

```
'/usr/oracle/dbs/newlog.dbf'
```

- Add the new log file and shutdown the database.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[2]
SVRMGR> shutdown
SVRMGR> startup mount <db_name>
SVRMGR> alter database
     2> add logfile '/usr/oracle/dbs/newlog.dbf'
     3> size 3M;
SVRMGR> shutdown
SVRMGR> exit
```

- Startup the database in restricted mode and check if the old logfile to be dropped is currently being used.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[2]
SVRMGR> startup open restrict
SVRMGR> select * from sys.v$logfile
     2> where member = '/usr/oracle/dbs/oldlog.dbf';
SVRMGR> select * from sys.v$log
     2> where group# =
              <Group# number retrieved from above query>;
```

---

2. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

- If the *STATUS* returned by the query is *CURRENT,* then run the following

```
SVRMGR> alter system switch logfile;
```

to switch to a different logfile, then drop the old logfile

```
SVRMGR> alter database
     2> drop logfile '/usr/oracle/dbs/oldlog.dbf';
SVRMGR> shutdown
SVRMGR> exit
```

- After a successful drop: first rename the dropped file; then shutdown and restart Oracle at least twice to make sure that no errors have been made; then the renamed drop file may be deleted.

Windows NT/2000 Move Log File

The following is an example showing a logical move of a log file named

```
'c:\Win32App\PVCS\Dimensions\orant\database\oldlog.dbf'
```

to

```
'c:\Win32App\PVCS\Dimensions\orant\database\newlog.dbf'
```

- Add the new log file and shutdown the database:

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[3]
SVRMGR> shutdown
SVRMGR> startup mount <db_name>
SVRMGR> alter database
     2> add logfile
'c:\Win32App\PVCS\Dimensions\orant\database\newlog.dbf'
     3> size 3M;
```

---

3. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

```
SVRMGR> shutdown
SVRMGR> exit
```

- Startup the database in restricted mode and check if the old logfile to be dropped is currently being used.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[4]
SVRMGR> startup open restrict
SVRMGR> select * from sys.v$logfile
     2> where member =
'c:\Win32App\PVCS\Dimensions\orant\database\oldlog.dbf';
SVRMGR> select * from sys.v$log
     2> where group# = <Group# number retrieved from
above query>;
```

- If the *STATUS* returned by the query is *CURRENT* then run the following:

```
SVRMGR> alter system switch logfile;
```

to switch to a different logfile, then drop the old logfile

```
SVRMGR> alter database
     2> drop logfile
'c:\Win32App\PVCS\Dimensions\orant\database\oldlog.dbf';
SVRMGR> shutdown
SVRMGR> exit
```

- After a successful drop: first rename the dropped file; then shutdown and restart Oracle at least twice to make sure that no errors have been made; then the renamed drop file may be deleted.

---

4. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

Rollback Segments

You may need to create new rollback segments in addition to the *SYSTEM* rollback segment and the four segments created at the installation of Dimensions. If there are other products using the Oracle database using tablespaces other than *SYSTEM*, then additional rollback segments will need to be created.

We recommend the following number of rollback segments for simultaneous Dimensions access:

No. of Concurrent Dimensions Users	Rollback Segments
Less than 16	4
16 to 32	8
X > 32	X/4 (up to a maximum of 49)

The diagnostic script *pcms_snap.sql* assists in determining whether the number of rollback segments should be increased. See "Diagnostic Scripts to Analyze your Database" on page 142.

Creating Rollback Segments of the Same Size

To create another rollback segment, login to an account that meets the following requirements:

- "UNIX Prerequisites" on page 45
- "Windows NT/2000 Dimensions Server Prerequisites" on page 45

and proceed as follows:

- Create rollback segment *rb6*.

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[5]
SVRMGR> create rollback segment rbs_6
     2> tablespace pcms_rbs
     3> /
SVRMGR> alter rollback segment rbs_6 online;
SVRMGR> exit
```

- Edit *<db_name>*

  You must also edit the parameter *ROLLBACK_SEGMENTS* in the *<db_name>* file for the database instance as described in "init.ora Parameters" on page 148.

- Query number and size of rollback segments

  The above will create a rollback segment called *rbs_6* in the *pcms_rbs* tablespace using the storage parameters defined for that tablespace. The table to query when trying to obtain the number and size of rollback segments is *DBA_ROLLBACK_SEGS* in user *SYS*.

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[5]
SVRMGR> select * from sys.dba_rollback_segs;
SVRMGR> exit
```

  All rollback segments should be the same size and this is ensured by the above procedure.

5. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

NOTE To avoid errors, before attempting to create additional rollback segments or changing the size as described later, you must calculate the new required size of the *pcms_rbs* tablespace. You can get the current size from the command Dimensions *SPAC* (page 57) and then use Dimensions *INCR* (page 62), if necessary, to enlarge the *PCMS_RBS* tablespace.

Increasing the Size of Rollback Segments

If it is necessary to increase the size of the rollback segments, this may be done by first using the *INCR* command (see "INCR – Increase the RDBMS Database Space" on page 62) to enlarge the *PCMS_RBS* tablespace. Then proceed as follows:

- Alter the *PCMS_RBS* tablespace storage parameters:

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[6]
SVRMGR> alter tablespace pcms_rbs
     2> default storage (initial 400K
     3> next 200K
     4> minextents 2
     5> pctincrease 0);
SVRMGR> exit
```

- Stop and restart the database, as described in "Starting and Stopping Oracle" on page 133, to bring the new storage parameters into effect for the rollback segments.

6. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

Alternatively, the size of the rollback segments may be increased by the following steps:

- Create a new tablespace (say *phone_dbs*) of appropriate size, and with suitable default storage parameters as described in "Tablespaces" on page 161. Remember that the default storage parameters will determine the rollback segment size.
- Before stopping and restarting the database, create new rollback segments and remove the old rollback tablespace by commands of the following form:

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[7]
SVRMGR> create rollback segment r11 tablespace phone_dbs
     2> storage (initial 400K next 200K);
SVRMGR> create rollback segment r12 tablespace phone_dbs
     2> storage (initial 400K next 200K);
SVRMGR> create rollback segment r13 tablespace phone_dbs
     2> storage (initial 400K next 200K);
SVRMGR> create rollback segment r14 tablespace phone_dbs
     2> storage (initial 400K next 200K);
SVRMGR> alter tablespace pcms_rbs offline;
SVRMGR> alter rollback segment r01 offline;
SVRMGR> alter rollback segment r02 offline;
SVRMGR> alter rollback segment r03 offline;
SVRMGR> alter rollback segment r04 offline;
SVRMGR> drop rollback segment r01;
SVRMGR> drop rollback segment r02;
SVRMGR> drop rollback segment r03;
SVRMGR> drop rollback segment r04;
SVRMGR> drop tablespace pcms_rbs;
SVRMGR> exit
```

7. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

- Edit the <*db_name*> file as described in "init.ora Parameters" on page 148 to replace the old rollback segments r01 to r04 by the new segments *r11* to *r14*.
- Stop and restart the database as described in "Starting and Stopping Oracle" on page 133.

Tablespaces

This section describes how to create new tablespaces and to move existing tablespaces from one device to another. To increase the size of a tablespace, use the *INCR* command described in "INCR – Increase the RDBMS Database Space" on page 62.

Initially all Dimensions tablespaces are created in the same directory (by default this is the *$ORACLE_HOME/dbs* directory (UNIX) or *orant\database* directory (Windows NT/2000)). Subsequently it may be advantageous to move some of the tablespaces to different devices. This provides an opportunity to balance up input/output loads or to make suitable space allocations for very large tablespaces.

Reasons for creating tablespaces are numerous:

- To separate project data from the Oracle data dictionary.
- To separate individual project(s) data – consider the use of a separate tablespace for different set(s) of Dimensions products.
- To separate all rollback activity from user data, preventing a single disk failure from causing permanent loss of data.
- To enable individual tablespaces to be taken offline while others remain online.
- To reserve a tablespace for a particular project.

- To enable image backups of individual tablespaces, while other tablespaces remain online.

Preliminaries

Before database administration operations can be carried out on tablespaces, it is necessary to change the rollback segments in use. The following are the required actions:

- Login to an account that meets the requirements of “also need to meet further operating-system-specific prerequisites" as described in the following subsections.
- Refer to “UNIX Prerequisites,” or “Windows NT/2000 Dimensions Server Prerequisites,” both on page 45.
- Shut down the database, and restart in restricted DBA mode (see “Starting and Stopping Oracle” on page 133).
- Take normal rollback segments offline, and bring *SYSTEM* rollback segments into use, as follows:

```
$ svrmgrl
SVRMGR> connect internal/<sys_password>[8]
SVRMGR> alter rollback segment r01 offline;
SVRMGR> alter rollback segment r02 offline;
SVRMGR> alter rollback segment r03 offline;
SVRMGR> alter rollback segment r04 offline;
SVRMGR> alter rollback segment r0 online;
SVRMGR> exit
```

8. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

NOTE Any extra rollback segments added as described in "The SQL*Plus scripts can be entered via various Oracle Command Line Interfaces – see "Access to the DBA Utilities" on page 46." on page 152 must also be taken offline.

Create New Tablespace

When a new tablespace is to be created, the user should first determine the purpose for which it is to be used, the required size and the device and directory in which it is to be placed. The user should also fix on a name for the tablespace and a filename. It is suggested that the tablespace name should reflect the purpose of the tablespace, and the filename should make reference to the tablespace name and to the relevant Oracle instance identifier (<*orasid*>).

- A tablespace to hold Dimensions tables is created with no storage clause, by commands of the following form (a UNIX example is shown):

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[9]
SVRMGR> create tablespace phone_data
     2> datafile '/disk1/dbfiles/phone_data_ora8i.dbf'
     3> size 80M;
SVRMGR> exit
```

9. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

- A tablespace to be used for users' temporary segments is created with a storage clause, by commands of the following form (a UNIX example is shown):

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[10]
SVRMGR> create tablespace phone_temp
     2> datafile '/disk2/dbfiles/phone_temp_ora8i.dbf'
     3> size 10M
     4> default storage (initial 500K
     5> next 500K
     6> pctincrease 0);
SVRMGR> exit
```

- A tablespace to be used for rollback segments is created with a storage clause, by commands of the following form (a UNIX example is shown):

```
% svrmgrl
SVRMGR> connect internal/sys_password>[10]
SVRMGR> create tablespace phone_rbs
     2> datafile '/disk3/dbfiles/phone_temp_ora7.dbf'
     3> size 30M
     4> default storage (initial 128K
     5> next 128K
     6> pctincrease 0
     7> minextents 2);
SVRMGR> exit
```

NOTE Do *not*, under any circumstances, place any database files on an NFS mounted disk.

---

10. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

Moving a Tablespace

A tablespace is moved from one device or directory to another by taking it offline, renaming it and putting it back online, by commands of the following form:

UNIX Operating System

- Take the tablespace offline.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[11]
SVRMGR> alter tablespace phone_temp offline;
SVRMGR> exit
```

- Copy the tablespace's operating-system file.

```
% cp /disk3/dbfiles/phone_temp_ora8i.dbf \
/disk99/db_temp_files/phone_temp_ora8i.dbf
```

- Rename the database.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[11]
SVRMGR> alter tablespace phone_temp
2> rename datafile '/disk3/dbfiles/phone_temp_ora8i.dbf'
3> to '/disk99/db_temp_files phone_temp_ora8i.dbf';
SVRMGR> exit
```

- Put the renamed tablespace back online.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[11]
SVRMGR> alter tablespace phone_temp online;
SVRMGR> exit
```

---

11. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

Windows NT/2000 Operating Systems

- Take the tablespace offline.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[12]
SVRMGR> alter tablespace phone_temp offline
SVRMGR> exit
```

- Copy the tablespace's operating-system file.

```
$ copy e:\database\phone_temp_ora8i.dbf
   f:\db_temp_files\phone_temp_ora8i.dbf
```

- Rename the database.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[12]
SVRMGR> alter tablespace phone_temp
   2> rename datafile 'e:\database\phone_temp_ora8i.dbf'
   3> to 'f:\db_temp_files\phone_temp_ora8i.dbf';
SVRMGR> exit
```

- Put the renamed tablespace back online.

```
% svrmgrl
SVRMGR> connect internal/<sys_password>[12]
SVRMGR> alter tablespace phone_temp online;
SVRMGR> exit
```

Bringing Tablespaces into Use

When one or more tablespaces have been created or moved as described above, they may be brought into general use by shutting down the database and doing a normal restart as described in "Starting and Stopping Oracle" on page 133.

---

12. Note that Oracle no longer supports the use of the *svrmgrl* command `connect internal` without the *SYS* password.

8 Backup Plan and Recovery after Media Failure

NOTE Please refer to the note in "Backup and Recovery" on page 122, regarding MERANT's prerequisites for assisting you in recovery from a database failure.

There is always a possibility that there will be a catastrophic disk hardware failure that results in the Oracle database or Dimensions libraries becoming unusable. Recovery from this situation will depend on the frequency and type of backups, the type of failure and also whether Oracle is run in *ARCHIVELOG* mode or not (contact PVCS Support Center for more information). The Database Administrator should determine precisely what backups are being taken and with what frequency. There should be a tried and tested recovery plan in place before the media failure occurs. The plan should identify that there is always a possibility of loss of data, which will involve users in having to re-input their work. The maximum time length of the loss should be quantified as a risk. It is important that your management and Dimensions Product Managers are made aware and accept the risk *before* the media failure occurs.

The extent of media failure needs to be accurately determined so that recovery is planned to achieve minimum data loss. As an extra precaution you can backup the disks immediately before any restore operations are started. A minor media failure can develop into a major loss of data if the wrong restore is carried out or there is a second media failure.

It is important to recover to a consistent set of the Oracle database and the Dimensions item libraries.

If the database and libraries are *all* on one disk then restoring a complete disk image will result in a consistent roll back with all the work since the *last backup* having to be re-entered.

If more than one disk is being used, then it will be necessary to study the situation more carefully. Individual files may require restoring to achieve consistency and again re-input of work will be required.

CAUTION! In the case of database or media recovery being required, then you must contact the PVCS Support Center before proceeding further.

A Default init.ora Provided by Dimensions

A MERANT Oracle runtime installation automatically tailors the Oracle set up parameters – defined in the UNIX or Windows NT/2000 file listed below – to appropriate values (in accordance with the number of concurrent users specified during the installation).

Operating System	Directory	File
UNIX	*$ORACLE_HOME/dbs/init<db_name>.ora*	*init<db_name>.ora*
Windows NT/2000	*c:\PVCS\Dimensions\ORANT\database*	*init<db_name>.ora*

B PFILE: Specifying the Oracle Initialization Parameter File

The *PFILE* option specifies the name of the initialization parameter file utilized in Oracle startup operations (at the operating system prompt, within the *SVRMGR* command-line interface or within the SQL command-line interface). If no *PFILE* command-line option is specified, Oracle uses the default *init<sid>.ora* file.

The following is the simplified syntax description for Oracle startup with regard to *PFILE*:

```
STARTUP [DBA] [FORCE] [RESTRICT}
   [PFILE=filespec] [EXCLUSIVE | SHARED]
   [MOUNT dbname | OPEN dbname] [NOMOUNT]
```